Fuzzy Optimal Control

by

Umar Iqbal Bhatti

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS ENGINEERING

November, 1997

FUZZY OPTIMAL CONTROL

BY

UMAR IQBAL BHATTI

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In

SYSTEMS ENGINEERING

NOVEMBER 1997

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN, SAUDI ARABIA

This thesis, written by

Umar Iqbal Bhatti

under the direction of his Thesis Advisor, and approved by his Thesis committee, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

Thesis Committee:

Chairman (Dr. M. Shahgir Ahmed)

Co-Chairman (Dr. Fouad M. Al-Sunni)

Member (Dr. M. A. El-Shafei)

Dr. A. A. Andijani
Department Chairman

Dr. A M. Al-Shehri
Dean College of Graduate Studies

Date: 7-12-97

Abstract

Name: Umar Iqbal Bhatti

Title: Fuzzy Optimal Control

Major Field: Systems Engineering

Date of Degree: November 1997

In this thesis a systematic scheme is proposed to design fuzzy optimal controllers. The proposed scheme adjusts the parameters of the fuzzy controller using a gradient descent scheme in order to optimize a chosen performance criterion. The optimization is carried out employing a gradient descent method. The gradient computation is performed using the BPD (Block Partial Derivative) technique. Simulation studies are reported for typical servo and regulator problems.

Master of Science Degree

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia

November, 1997

خلاصة الرسالة

الإسم: عمر إقبال بهتي

العنوان: تحكم غامض أمثل

التخصص: هندسة نظم

تاريخ الشهادة: نوفمبر 1997

نقترح في هذه الرسالة تصميم متحكمات مثلى معتمدة على المنطق الغامض. الطريقة المقترحة لتصميم المتحكمات تعتمد على تغيير معاملات المتحكم بهدف الوصول إلى أفضل أداء.إن هذا التغيير في المعاملات يعتمد على إيجاد مشتقة دالة الأداء ,ولإيجاد هذه المشتقة نستخدم طريقة المشتقة الجزئية للمجموعات و للتحقق من حسن أداء المتحكم المقترح نقوم بدراسةٍ تعتمد على محاكاة بعض الأنظمة الديناميكية.

ماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران- المملكة العربية السعودية

نوفمبر 1997

Dedicated to

My Late Father

Prof. Muhammad Iqbal Tahir

Acknowledgements

All praise be to Allah, the Lord of the worlds. May peace and blessings be upon Mohammad, the last messenger, and his family. I thank Allah for His limitless help and guidance at every instant in my life.

I would like to express my deep and heart-felt gratitude and appreciation to Dr. M. S. Ahmed, my thesis advisor, for his valuable advise and guidance throughout this research. It is an indeed an unforgettable experience to work with him to learn much beyond academics.

I would also like to thank the other committee members: Dr F.M. Al Sunni and Dr M. A. ElShafei for their consistent support and co-operation.

I acknowledge the help and support rendered by King Fahd University of Petroleum & Minerals for this research.

Lot of gratitudes to all my friends and colleagues who have helped, encouraged and beared with me especially Shariq, Noman T., Azfar, Abdul Rahim, Noman K., Izzat, Adnan, Aijaz, Anas, Hasan, Imran, Asad, Shuja, Rizwan, Rais, Mukarrum, Shahid, Arif, Sohail, Rashid and Uvais.

Finally, special thanks to my parents, family members and siblings who were always there for me and who sacrificed their own welfare for mine.

Contents

List of Figures

List of Tables

Nomenclature

FLC Fuzzy Logic Controller

FMRLC Fuzzy Model Reference Learning Control

ANFIS Adaptive Neuro based Fuzzy Inference System

BP Back Propagation

TD Temporal difference

CMAC Cerebellar Model Articulation Controller

GARIC Generalized Approximate Reasoning Based Intelligent Control

GA Genetic Algorithm

BPD Block Partial Derivatives

DFL Dynamic Focus Leaning

BPTT Back propagration through time

DB Dynamic Backpropagation

e error between the output of the control system and the input command

u input applied by the controller to the plant

y output of the closed loop control system

J objective function

μ^l representation of the centroids of the input membership functions

σ^l variance of the input membership function

$bias$ link at the output of the Fuzzy Logic controller with input unity and have trainable weight.

n_{mf} number of input membership functions used

y^l centroids of the output membership functions

mf representation of the membership function

a weighted sum of the output of the multipliers in the Fuzzy Logic controller

b sum of the output of the multipliers of FLC

Δe change in error at each iteration

n number of inputs in the FLC

M value of number of membership function raised to power the number of inputs

z^l output of the multipliers of the FLC

Δu output of the FLC

C time average polynomials used in the objective function with the error term

D time average polynomials used in the objective function with the control effort term

H penalty on the control effort term in the objective function

t time variable

V power of the control effort term penalty

k number of time steps for which the system is simulated

k_e scaling factor for error

k_{de} scaling factor for change in error

k_y scaling factor for the output

k_u scaling factor for the control input

e_f error multiplied by scaling factor

y_f output multiplied by its scaling factor

Δe_f change in error multiplied by its scaling factor

u_c command input

y^l the output centroids

Δu_f the output of the FLC multiplied by its scaling factor

Chapter 1

Fuzzy Logic Control

In control systems, one is confronted with the error between a system output and a reference command. This error is usually represented by a crisp value. However in fuzzy terms which attempts to imitate the action of a human operator this error may be classified as 'large', 'small' or 'very small' error. These variables are termed as fuzzy variables. Human operator engaged in controlling a system employs this sort of information. For example consider the case of balancing a stick. A human can balance it by monitoring the error of the stick from being straight. If he found it to be tilted he will move his hand along the tilt. This idea replaces the conventional steps of modelling and designing control for complex systems.

1.1 Introduction

Fuzzy Logic Control (FLC) is used extensively nowadays. It is based on human intuition and experience. Traditionally heuristics and experience of operators form the basis in process control. Unlike other formal control techniques this human knowledge could be incorporated in the FLC. Fuzzy Logic introduced by Zadeh [1] is based on linguistic variables that do not have precise values. For example "small" or "large" could be termed as linguistic variable.

In general, control systems are designed by feeding back a crisp error between the reference command and the process output. But in case of a human operator crisp value of error is not needed. This leads to the basis of using Fuzzy Logic. Operator can express fuzzy control action based on fuzzy error. This process is translated to a Fuzzy Logic controller using the Fuzzy Logic theory.

Typical FLC designs are done by trial and error. The controller parameters are adjusted using designer's experience and intuition until a satisfactory response is obtained. Usually no performance criterion is used to grade the performance. As a result, the conventional design method, however better

control it gets, could not be termed as optimal.

In the proposed method an optimal scheme is presented to improve the FLC design. In this approach, an initial design of FLC based on the plant's response is obtained by trial and error. Next in order to design an optimal, a performance index is selected. The FLC parameters are tuned to minimize this criterion.

The optimization of the controller is done using a gradient algorithm that iteratively updates the FLC parameters in a batch mode, while the process is subjected to all possible conditions it is likely to experience during its implementation. The gradient computation is carried on using the back propagation (BP) in conjunction with the recently proposed Block Partial Derivative (BPD) technique.

1.2 FLC design

Fuzzy Logic Control (FLC) is based on the response, knowledge, experience and reasoning of human in controlling a process. Fuzzy variables are defined explicitly and the control is based on a set of logical statements based on these variables.

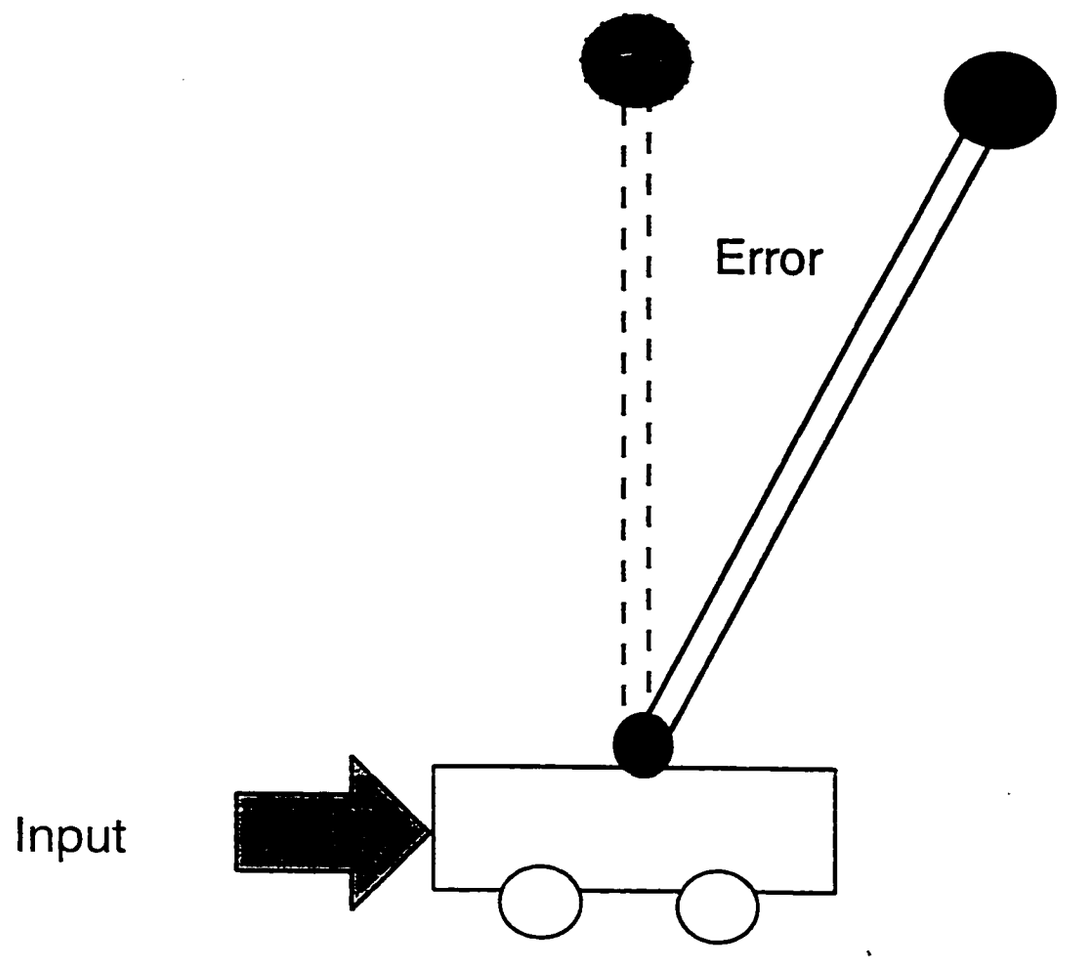


Figure 1.1: The inverted pendulum system: The case when the error is LARGE

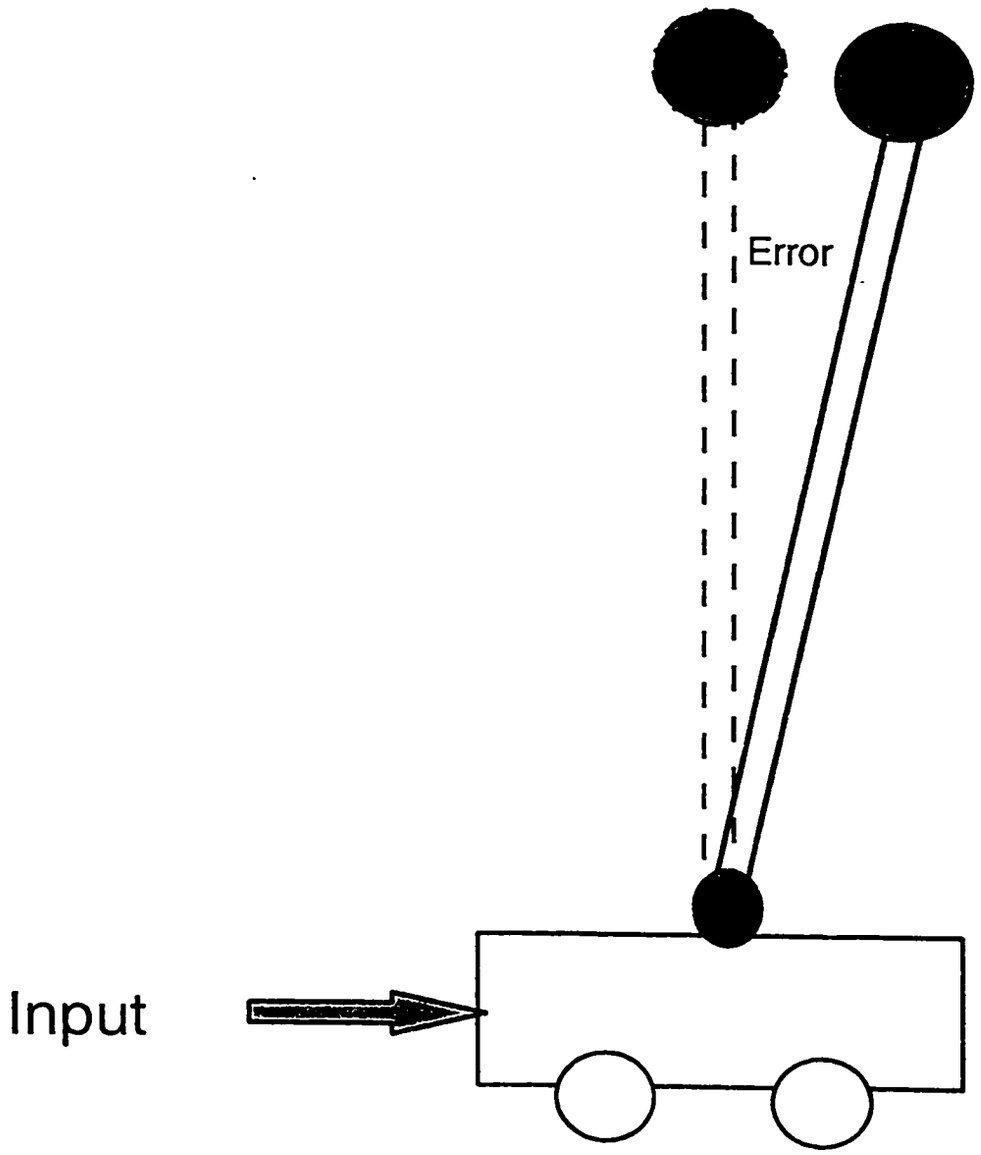


Figure 1.2: The inverted pendulum system: Situation when the error is SMALL

For example in the case of a typical control system i.e inverted pendulum (shown in Figures 1.1 and 1.2) , the following rules could be defined

IF error is LARGE input is LARGE

IF error is SMALL input is SMALL

.....

As in the case of typical control system, shown in Figure 1.3, a controller uses the error to control the process. FLC constitutes a part of that controller.

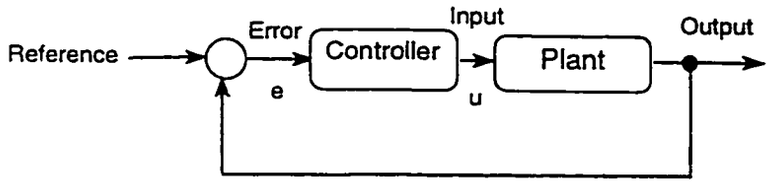


Figure 1.3: Typical control system

The basic elements of a fuzzy control system can be described by the block diagram shown in Figure 1.4. As shown in the figure. the fuzzy control system is composed of many functional blocks.

In the fuzzifier block the crisp variables or numbers are changed to fuzzy variables. This block uses membership functions. A set of membership functions known as triangular membership functions is shown in Figure 1.5.

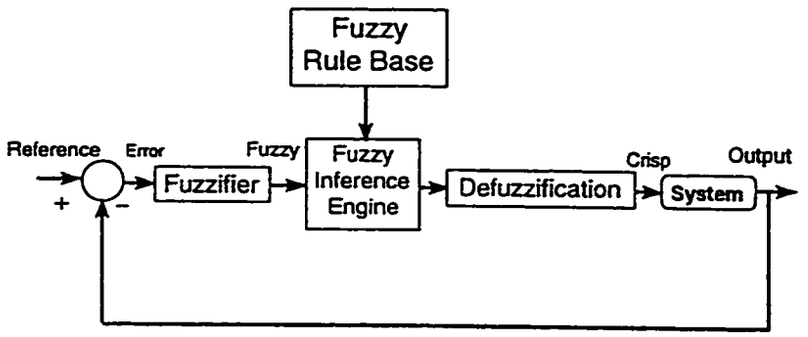


Figure 1.4: Block Diagram of the Fuzzy Logic control System

There could be other membership functions like the sine and gaussian membership functions.

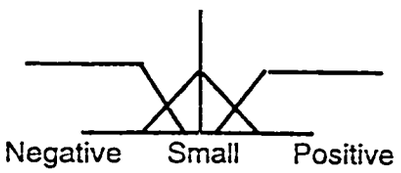


Figure 1.5: A Triangular Membership Function set

Using crisp variables as input, a fuzzifier block assigns membership functions to the fuzzy variable. For example, assume that a crisp number is positive and larger than the small membership function range. Using the membership functions of Figure 1.5, it will then be assigned as Positive fuzzy

variable with membership value ranging from 0 to 1 depending on its magnitude. These fuzzy variables are then passed through the fuzzy inference engine. This engine uses fuzzy rule base for its operation. Decision is made on the fuzzy variables employing statements like

IF error is NEGATIVE and change in error is NEGATIVE use POSITIVE input

It should be noted here that this statement uses the fuzzy values of error and change in error instead of the crisp values to decide to use a POSITIVE input. This decision uses the rule base i.e POSITIVE is defined in the rule base. Rule base contains actions for different fuzzy inputs. Most common rule base is in the form of a lookup table formed by the designer.

As we need a crisp value for the control action, a defuzzification block is needed next. It changes this fuzzy variables to a crisp control input. This input is then used to drive the plant.

1.3 Takagi Sugeno Fuzzy system

After the informal introduction in the preceding section it will be very much informative to introduce Takagi Sugeno fuzzy system. It is the most common

fuzzy system in use as presented by Takagi and Sugeno [9]. The block diagram of the system is given in Figure 1.6. A Multiple input single output (MISO)

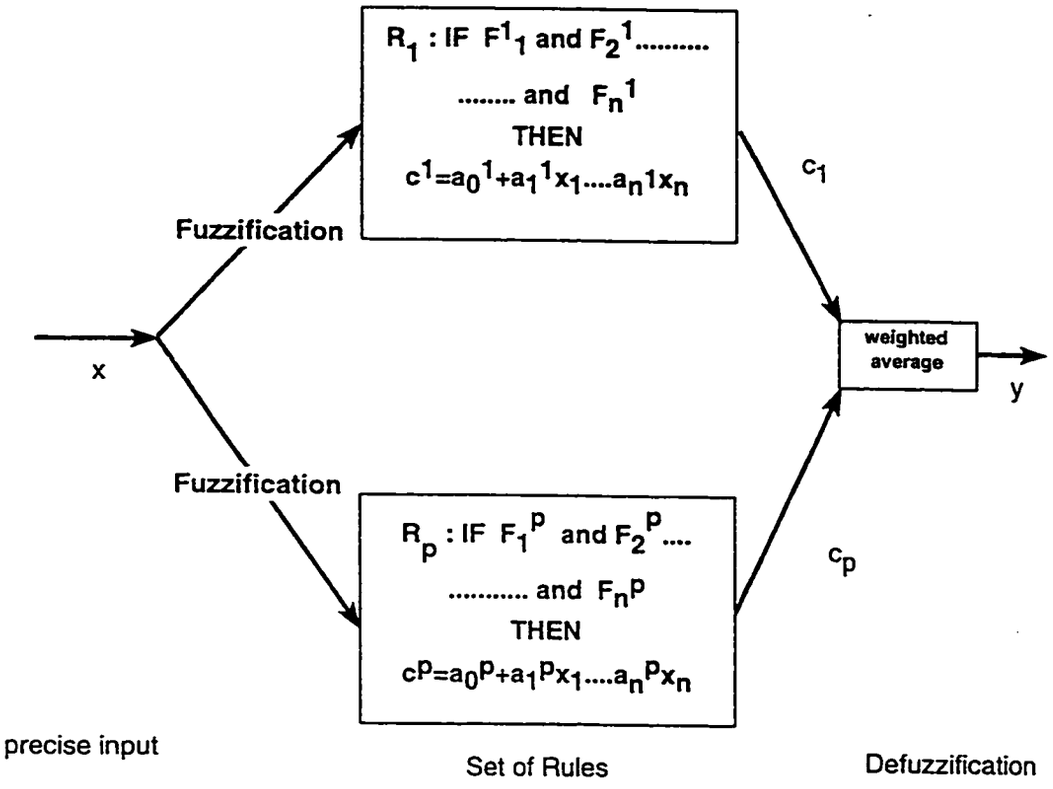


Figure 1.6: Takagi Sugeno Fuzzy system

fuzzy system could be defined as a non-linear mapping from an n dimensional

input vector

$$X = [x_1, x_2, ...x_n]^T$$

to a scalar output y. The fuzzy system could be represented by a set of p IF-THEN rules, that could be expressed as

R_1 : *If* ($\overline{x_1}$ *is* $\overline{F_1^1}$ *and ... and* $\overline{x_n}$ *is* $\overline{F_n^1}$) *Then* $c_1 = g_1(X)$

.....

.....

R_p : *If* ($\overline{x_1}$ *is* $\overline{F_1^p}$ *and ... and* $\overline{x_n}$ *is* $\overline{F_n^p}$) *Then* $c_p = g_p(X)$.

$\overline{x_b}$ is the output of membership function for the input x_b hence $\overline{x_b}$ is a fuzzy variable. Premise $\overline{F_b^a}$ is the a^{th} linguistic value associated with linguistic variable $\overline{x_b}$. And $c_q = g_q(X)$ is the consequence of the q^{th} rule using IF statements . The above expression using the fuzzy set theory can be written as

R_1 : *If* (F_1^1 *and ... and* F_n^1) *Then* $c_1 = g_1(X)$

.....

.....

R_p : *If* (F_1^p *and ... and* F_n^p *Then* $c_p = g_p(X)$

where F_b^a is a fuzzy set defined by $F_b^a := \{(x_b, \mu_{F_b^a}(x_b)) : x_b \epsilon R\}$. The mem-

bership function, $\mu_{F_b^a} \epsilon [0,1]$ quantifies how well the linguistic variable $\overline{x_b}$ represents input x_b. The output of this relation is the linguistic value F_b^a .

In this model, multiplication is used to represent *and* operation. While *min*(minimum) function could also be used as presented in Takagi and Sugeno [9]. In the case of use of multiplication, the problem of discontinuity of function does not occur. Defuzzification may be obtained using

$$\overline{y} = \overline{f(X)} = \frac{\Sigma_{i=1}^{p} c_i (\Pi_{j=1}^{n} \mu_{F_j}(x_j))_i}{\Sigma_{i=1}^{p} (\Pi_{j=1}^{n} \mu_{F_j}(x_j))_i}$$

Where p is the number of rules and n is the number of membership functions used.

It is assumed that the denominator is not zero so the value of the output does not become infinite. The above equation could be expressed as

$$\overline{y} = c^T \eta \text{ where}$$

$$c^T := [c_1 ... c_n]$$

and

$$\eta^T := \frac{[(\Pi_{j=1}^{n} \mu_{F_j}(x_j))_1 ...]}{\Sigma_{i=1}^{p} (\Pi_{j=1}^{n} \mu_{F_j}(x_j))_i}$$

The vector c could be a function of the inputs. However for simplified and most common case c_i are centers of the output membership functions provided by the experts. In this model as shown by Wang [34] the decision part

is not fuzzy. The improved model presented by Wang [34] however takes the same form finally but the decision part is fuzzy. This facilitates the incorporation of Expert knowledge in the controller. There are also other methods of defuzzification available, as presented by Wang [34].

Chapter 2

Literature Review

Fuzzy logic was introduced by Zadeh [1] in 1965. Applications to control theory sprung up in the 70's when Mamdani *et al.* [2][3] applied fuzzy logic in process control. Although fuzzy control has been in use for about a decade it was only recently that Kosko *et al.* [5] and Wang *et al.* [42] provided the mathematical foundation by showing that fuzzy systems are universal approximators. A large number of applications in the decade of 80 revolutionized the market especially in the domain of the consumer products as mentioned by Schwartz *et al.* [6]. Chuen [7] and Piero *et al.* [8].

By the advent of 90's the trend of combining fuzzy logic with the neural networks started receiving attention. Most of the approaches based their

work on the Takagi Sugeno model [9] which is essentially a generalization of the Fuzzy Logic concept.

2.1 General Training Approaches

The idea of fusing the fuzzy logic and neural networks was presented by Gupta *et al.* [10] and Ching *et al.* [12][13]. The learning of the FLC both off-line and on-line began on a large scale in the nineties. Ayoubi [33] proposed a method to train the fuzzy network using the backpropagation (BP) in which the desired training set is needed as a prerequisite.

Arabshahi [35] introduced fuzzy methods for the acceleration of back-propagation. The FLC is trained in order to imitate input output data set.

Ollennu *et al.* [37] used the input output linearization method with an outer loop consisting of an adaptive fuzzy network. The fuzzy output is used as a supporting input to the adaptively generated output and is used for cancelling the effect of disturbances. The fuzzy structure is 'identified' adaptively in the work. The linear feedback control law is used in the inner loop. The approach is quite restricted and is difficult to modify for other applications. Also this approach is limited to non-linear systems which are

feedback linearizable.

Tanaka *et al.* [20] used a fuzzy identified model for the replacement of the plant dynamics. The error BP through the fuzzy identified network is used for training the controller.

Spooner *et al.*[23] presented adaptive control techniques for non-linear systems using fuzzy systems and Lyapunov theory. However the treatment is restricted to feedback linearizable plants.

Chen *et al.* [27] applied H_∞ tracking design to the FLC but the method is applicable to the systems which are linear in control. The algorithm is quite complex. Robustness is claimed over a range of disturbance inputs.

Watanabe *et al.* [39] used FGNN (Fuzzy Gaussian neural networks) for the control of mobile robot. For the training, a delta rule (back propagation) is used. However no systematic method of error backpropagation through the plant is presented.

2.2 Reinforcement Learning

In reinforcement learning, a signal called reinforcement signal is produced. It is generated as an estimation of the performance of the process. This may

be a function of states or a statistical measure.

Berenji *et al.* [29] presented the algorithm known as GARIC (Generalized Approximate Reasoning Based Intelligent Control) for reinforcement learning of the FLC. The idea of three different combination of schemes is used to generate the control input. Two learning networks and one stochastic generator is used. One network is the conventional fuzzy logic controller that is trained using the steepest descent and back propagation algorithm. The other network evaluates the system states to generate a reinforcement signal. It is used as an objective function by the fuzzy logic network. These two signals are combined in the stochastic unit which uses normal distribution to generate the control signal. For the back propagation of the error through the plant, reinforcement signal and its approximate derivatives are used . Berenji *et al.*[44] applied this algorithm to control space shuttle attitude problem.

Lin *et al.* [12] proposed a fuzzy logic learning scheme including two phase learning. In the first phase using the input output data, fuzzy logic membership functions are initialized by a method analogous to statistical clustering. This phase replaces the rule defining phase that is usually completed by experts. In the second phase using error BP (steepest descent method) the controller is trained. However for the implementation of this scheme, the

controller input output data set must be available.

Lin *et al.* [13] proposed structure/parameter learning for FLC by optimizing specific performance criterion. A reinforcement signal from the environment is used as the objective function. This signal is designed to be maximized for better control. Reinforcement signal is obtained through statistical derivation. A fuzzy structure is proposed for the generation of this reinforcement signal in order to speed up learning. Similarly temporal difference (TD) method is used to remove the delay in learning. TD method has the advantage of learning using prediction in between the iterations when data samples are not available. The structure and parameters of the FLC both are tuned. However instead of direct error BP, predicted value of the error is used.

Lin *et al.* [28] continued the research devising improved algorithms incorporating linguistic teaching signals for learning. Provision of condensation/ reduction of the fuzzy laws is also presented. The main difference from the previous approach is in the reinforcement signal generation. Another fuzzy structure is used for this task. However the approach suffers the problem of absence of direct error signal through the plant for BP. Also it becomes too complex due to the use of more fuzzy architectures. Further, in order

to get satisfactory performance, prohibitively long experimentation phase is generally required.

2.3 Fuzzy CMAC Learning

CMAC stands for Cerebellar Model Articulation Controller. Functionally a trained CMAC is equivalent to a lookup table. Structurally it consists of three layered architecture. Essentially this is a type of neural network.

Nie *et al.*[14] modified the CMAC nets incorporating fuzzy logic in them. Learning method used is competitive. It means; if objective function improves, structure is preserved otherwise it is modified. For getting objective trajectory a general algorithm with many variants is used. From a reference plant an initial target trajectory is used to train and restructure FCMAC.

2.4 FMRLC

FMRLC stands for Fuzzy Model Reference Learning control. Procyk [4] presented learning of FLC based on reference model. The deviation from the actual response is termed as a variable which is used as an indication of the

correction required in the control action. A correction table is formed and fuzzy rule base is updated using the table. The corrections are computed from the inverse of plant model which is identified on-line. This technique is not applicable in case of NMP (Non-Minimum Phase) plants. The generation of reinforcement signal is not performed systematically in this scheme.

Yin *et al.* [16] used on-line and off-line methods for training of fuzzy controller in the implementation of MRAC. First the off-line membership parameters are learned for initial estimated system parameters. Then the controller parameters are updated to reduce error between the plant and the model . This process is continued until the error is below a criterion. The adaptation law for the control input is derived using the Lyapunov theory.

Layne [19] and Mougdel [30] proposed FMRLC in which the learning is based on the information of the plant inverse dynamics. Essentially the desired output is presented to the inverse fuzzy model of the plant and the fuzzy controller inputs are generated. These inputs are combined with the control input to produce new control action. This is a two phase procedure. In the first phase it is observed which rules were fired and in the second phase they are altered. These are incorporated in the original FLC inputs. This approach does not have the facility of the modification of membership func-

tions. Kwong *et al.* [21] presented modification of FMRLC termed as DFL FMRLC (DFL stands for Dynamically Focussed Learning). This approach essentially trains the lookup table using the inverse fuzzy network (similar to that presented by Layne [19]). The table is modified dynamically in order to concentrate on that region of the table where more control rules are fired. Kwong *et al.* [24] used the same approach for fault tolerant aircraft control.

Hessburg *et al.*[22] developed MRAFLC (Model Reference Adaptive Fuzzy Logic Control) for vehicle guidance using a Lyapunov stability function. Stability conditions are derived and error BP term is formulated under the assumption that the closed loop system is bounded . In the Lyapunov treatment state space model of the plant is used.

2.5 Genetic Algorithm Learning

Use of GA (Genetic Algorithm) is a relatively recent trend in optimization. Linkens *et al.* [25] used GA to optimize an FLC. The essence of the method is that first the FLC structure is formed . Then using the GA coding, the FLC is represented in the form suitable for the application of GA. For example in the case of a triangular membership function three points are needed to specify

it . These codes are then optimized by the GA method i.e the reproductions and crossovers. The algorithm is restricted to the class of processes that are linear in control input. Also the performance of GA depends on the capacity of hardware used. Park *et al.* [17] used genetic algorithm with fuzzy logic. Krishnakamar *et al.* [38] used GA FLC for flight control. Karr *et al.* [36] also used Genetic Algorithms to train fuzzy logic controllers. Although this on-line algorithm is found to work on specific examples, stable performance is not guaranteed for the general case.

2.6 Fuzzy control with sliding mode design

Ohtani *et al.* [32] proposed sliding mode control of manipulator using fuzzy rules. The sliding mode gain is taken from a fuzzy set of rules. These rules are then changed empirically after observing the response. An empirical set of rule changes is presented. The method lacks a systematic method of tuning.

Vaqar [43] used Fuzzy Logic in sliding model control. It was mainly used for the controlling of chattering. Fuzzy logic proved its utility to handle complex and ill defined systems. However a predefined table of fuzzy rules is used to get the desired response.

2.7 ANFIS

ANFIS (Adaptive Network based Fuzzy Inference System) was proposed by Shing *et al.* [40]. For the training of the fuzzy network TBP (Temporal Back Propagation) is used. TBP was originally proposed by Derrick *et al.* [18]. For the error backpropagation a simplified difference equation model of the plant is used. Extending the approach, Shing *et al.* [41] proposed an error back propagation by calculation of Jacobians of system model. A large set of initial conditions is necessary for the training of the ANFIS using TBP.

Chapter 3

Fuzzy Network Training

Similar to neural networks, fuzzy networks could be trained to minimize certain performance index. Such training may be gainfully used to enhance the performance of FLC. Backpropagation (BP) technique normally is used for the training. The parameter updating algorithm generally adopted is one of the most common optimization methods i.e steepest descent rule or the gradient rule. The algorithm is based on the following iteration :

$$w_{t+1} = w_t - \eta \frac{\partial J}{\partial w_t} \tag{3.1}$$

Where w_t is the value of fuzzy parameter at instant t, η is a positive constant known as learning rate and $\frac{\partial J}{\partial w_t}$ is the gradient of the criterion function w.r.t

the fuzzy parameter values at the time instant t. Criterion function or the performance index J is usually taken as:

$$J = \frac{1}{2}\Sigma||e||^2 \tag{3.2}$$

Where e is the error between the output of the network and the desired output and is given by :

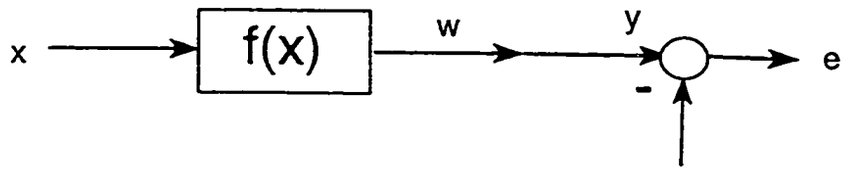
$$e = y_d - y \tag{3.3}$$

Where y_d is the desired output while y is the output of the Fuzzy Network (FN). Since y_d is independent of the fuzzy weights $\frac{\partial J}{\partial w_t}$ is given by :

$$\frac{\partial J}{\partial w_t} = -\frac{\partial y}{\partial w_t}||y_d - y|| \tag{3.4}$$

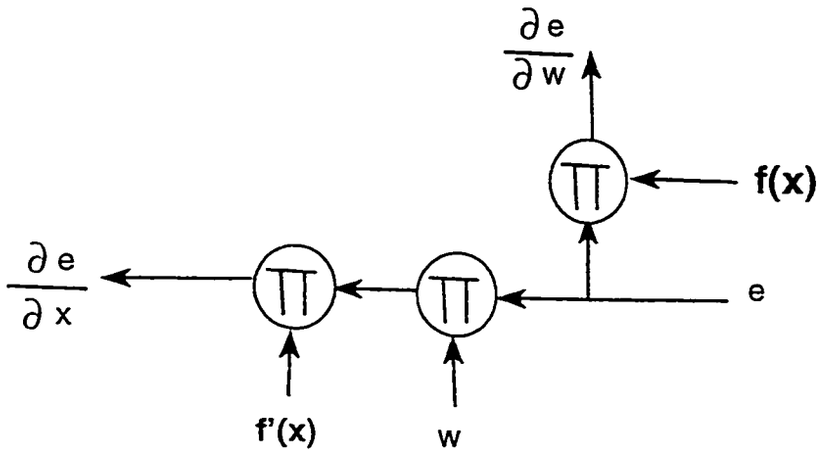
$\frac{\partial y}{\partial w_t}$ can be computed using the backpropagation.

The concept is illustrated through an example. Consider the case of a single neuron network shown in Figure 3.1(a). A neuron is a block characterized by a function and a link that is specified by certain weight. The output of the neuron is obtained by multiplying the weight of the link by the function evaluated at the input of the neuron. BP (Backpropagation) technique may be used to update the weight of the link to minimize the objective function as the one in Equation 3.2. BP diagram is shown in Figure 3.1 (b).



a) Forward Pass

$$y = w\, f(x)$$



b) Backward Pass (⊓ indicates multiplication)

Figure 3.1: Neuron Training a) Forward Pass b) Backward Pass

In order to apply BP technique for updating the neuron weight, essentially two passes are required. In the first pass, usually termed as the forward pass, the input is fed to the neuron and the output is computed. This output is used to calculate the the error and the objective function J. The second pass is the backward pass. In this pass gradients of J with respect to input as well as with respect to neuron weights are calculated. Using Eq 3.1 a new value of the weight is calculated. This constitutes one iteration. Similar iterations are continued until J goes below a certain specified threshold which is usually a very small number. The learning rate η in Equation 3.1 could take different values, which will be described later in the Section 3.7.

Although the idea presented here describes training of a single neuron, in practice training is needed for a network of neurons. A neural network usually has several layers of neurons containing many neurons in each layer. The layers are connected in a prescribed fashion. Fuzzy network has a similar structure. Therefore BP can also be applied to update fuzzy parameters.

As mentioned in the literature review, a large class of training approaches involve the steepest descent algorithm. The application of the algorithm needs the computation of the gradients. Although computation of gradient in the simple neuron of Figure 3.1(a) is straightforward, such computations

in complex networks may be quite involved.

In control systems, complex networks involving feedback loops and delay elements are generally encountered. Consider the example of a typical control system as shown in Figure 3.2. It is desired that the output of the system should follow the command. If a set of values for e and u are available, the controller can be trained to perform as a lookup table. Consider the case

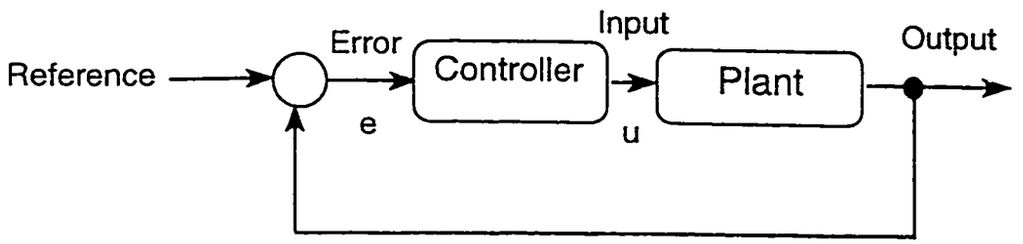


Figure 3.2: A typical control system

when the controller is to be replaced by a fuzzy logic controller (FLC). The FLC parameters can be updated by a gradient scheme such as in Equation 3.1. In such a case, gradients could easily be computed employing Equation 3.4. BP scheme can be applied for computing the gradients of FLC output with respect to its parameters. This is a typical neural network training as could be seen in Rumelhart [49].

When such training data (i.e set of e and u) is not available, the problem

becomes much more profound. In typical control problems only the desired output, plant model and the command input are given. By suitably choosing criterion function, the problem could be posed as an optimization problem. However, the difficulty arises in calculation of the required gradients. In the literature a few approaches are used to solve this problem. Some of the typically used schemes are BPTT (Back Propagation Through Time) presented by Werbos [50], DB (Dynamic Back Propagation) presented by Narendra [51,52] and BPD (Block Partial Derivatives) developed by Ahmed [31]. BPD scheme has certain advantages over the earlier two mentioned schemes, summarized as follows

1. In contrast to the application of chain rule for computing the gradients through the control system, BPD uses much simpler Mason Gain formula that eliminates a lot of cumbersome computations.

2. As BPD works by structuring the network into independent blocks, the computation can be done in a structured way. Every block could be handled independently and then these could be put together.

3. When BPD technique is applied, there is no restriction on the nature or number of blocks that could be used in the network except that these

are to be first order differentiable.

Although BPD is an extended form of DB, it does not suffer from the problem of complex derivation and employment of the chain rule. For complex control systems having feedback connections BPD is a handy tool to compute the gradients. BPD concept is explained in the following section.

3.1 Block Partial Derivatives (BPD)

Definition : Consider a block A as shown in the Figure 3.3 (a) with input $x \in R^{n_x}$ and output $y \in R^{n_y}$, the block partial derivative (BPD) for the block is defined as :

$$\frac{\partial^A y_i}{\partial x_j} = Lt_{\Delta x_j \to 0} \frac{\Delta y_i}{\Delta x_j}|_{\Delta x_k = 0} \quad \forall \quad k \neq j \tag{3.5}$$

Remark 1: In the above definition all inputs other than x_j are held constant. However, the other signals within the block are allowed to vary owing to the change in x_j.

Remark 2: If a parameter set $w \in R^{n_w}$ in the block also is allowed to vary, its effect may be accommodated by representing the weights as auxiliary inputs as shown in Figure 3.3 (b). This representation facilitates computation of BPDs with respect to the parameters.

Remark 3: When x and y are functions of time and the block is dynamic (i.e has memory) the BPDs may be represented by rational dynamic operators.

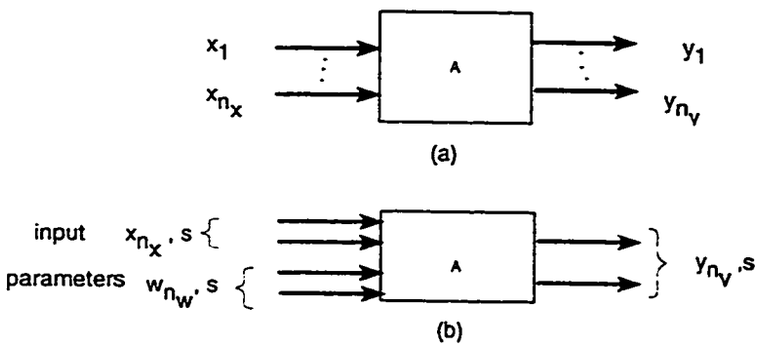


Figure 3.3: Representation of the BPD

3.1.1 Computation of the BPD's

A typical application of BPD is to compute gradients for a complex network. For example let the block A of Figure 3.4 consists of sub-blocks B, C, D and E. Each of the sub-block could have parameters to be updated. For application of the steepest descent rule, we need the gradient of the output with respect to the parameters of each of the sub-block as shown in Equation 3.4. These gradients are computed as follows: Each of the sub-block is replaced

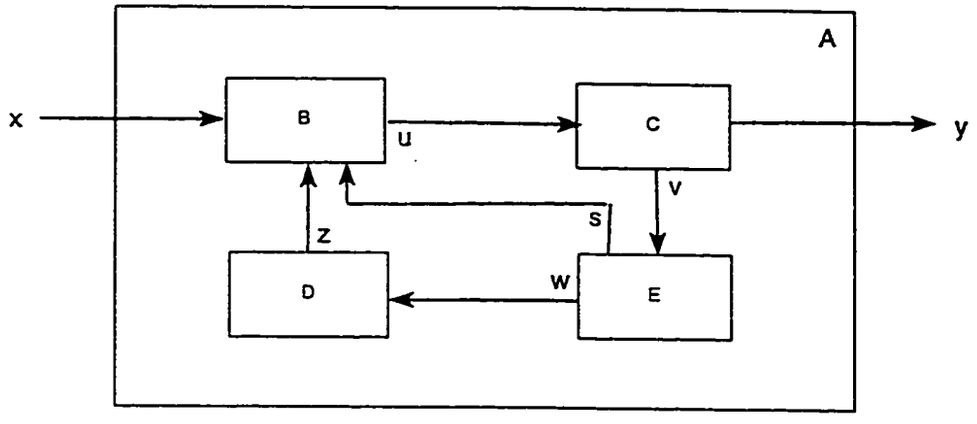


Figure 3.4: Division of the system into blocks

by its linearized equivalent. Linearized equivalent is essentially a linearized relationship between the input, output and adjustable parameters , in terms of its BPD's. If the block contains more than one input or parameters, each BPD constitutes a separate linear block and then these block outputs are added.

For simplicity, let us assume that only block B has an adjustable parameter b. The linearized equivalent of block A is shown in Figure 3.5. It could be seen that linearized equivalent for sub-block B has four linear blocks because of its three inputs and one adjustable parameter. The required gradient

needed for updating the parameter b is computed by applying Mason Gain Formula treating ∂b as input, ∂y as the output and the linearized equivalents as usual blocks in a conventional block diagram.

3.2 Application to FLC

In order to train the FLC, following steps are used (refer to Figure 3.6).

1. The output of the system is compared with the output of reference model to calculate the error.

2. Using the error, objective function value is obtained.

3. By applying BPD technique, gradients of the objective function with respect to the FLC parameters are computed.

4. The batch steepest descent algorithm is used to update FLC parameters.

5. The value of objective function is compared with the criterion, if it is not below the chosen criterion the whole process is repeated.

The inclusion of the reference model has the advantage that it can incorporate all the desired characteristics like the overshoot, rise time, settling

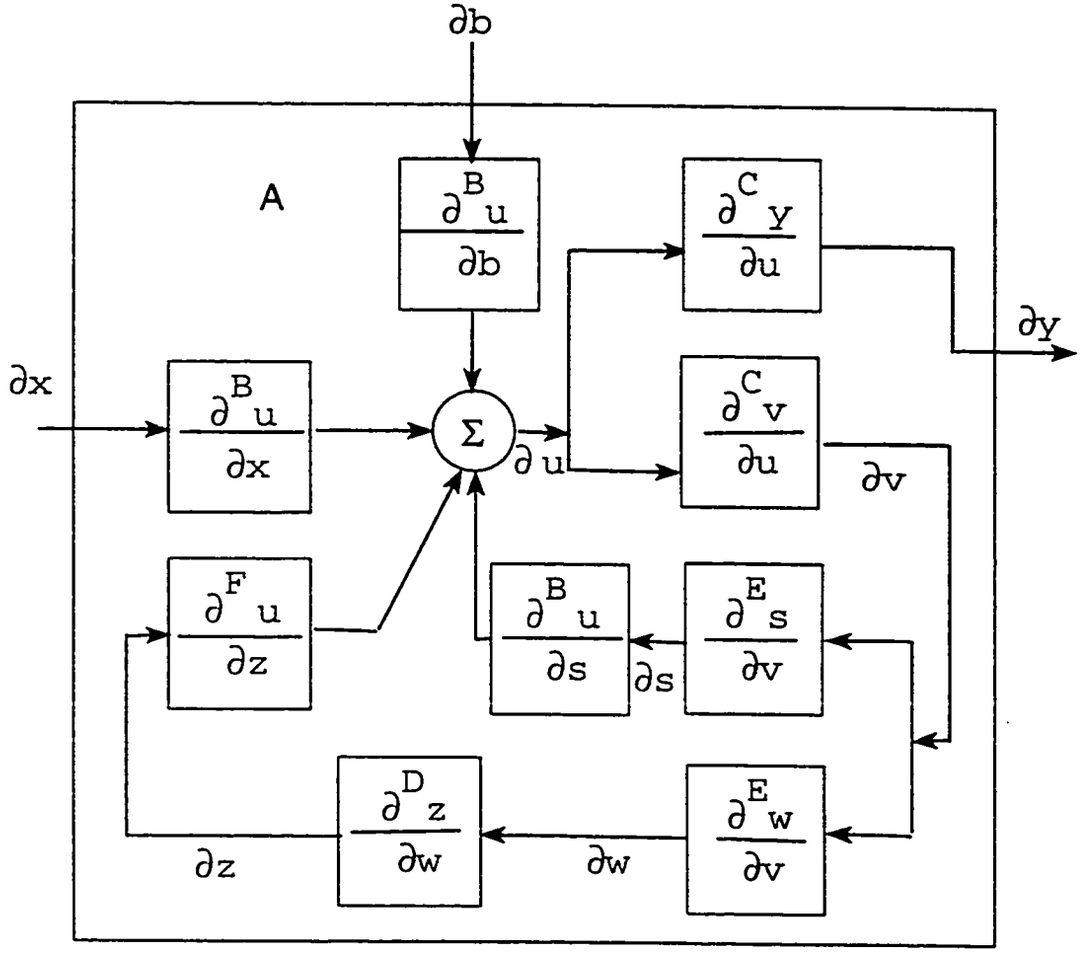


Figure 3.5: Linearized Network for BPD computation

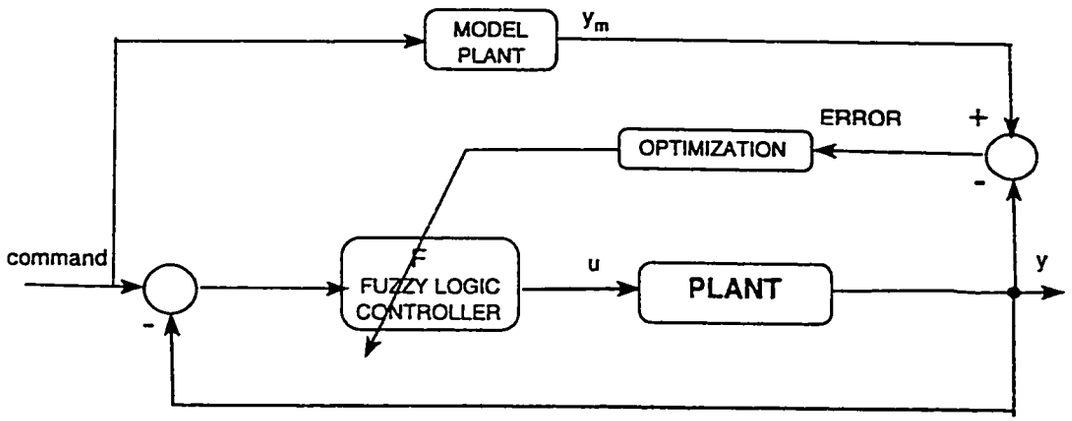


Figure 3.6: Closed Loop control system

time etc. to be followed by the plant.

Before the presentation of the network training details, FLC configuration will be discussed in the next section.

3.2.1 Controller Structure

A Fuzzy Logic controller is shown in the Figure 3.7. The inputs to the FLC are the error between the command input and the output of the system e, the change in this error over a time instant Δe and the plant output y. In the conventional FLC, the plant output is not used as input to the FLC. However, in the proposed scheme this addition is made in order to make the fuzzy rules

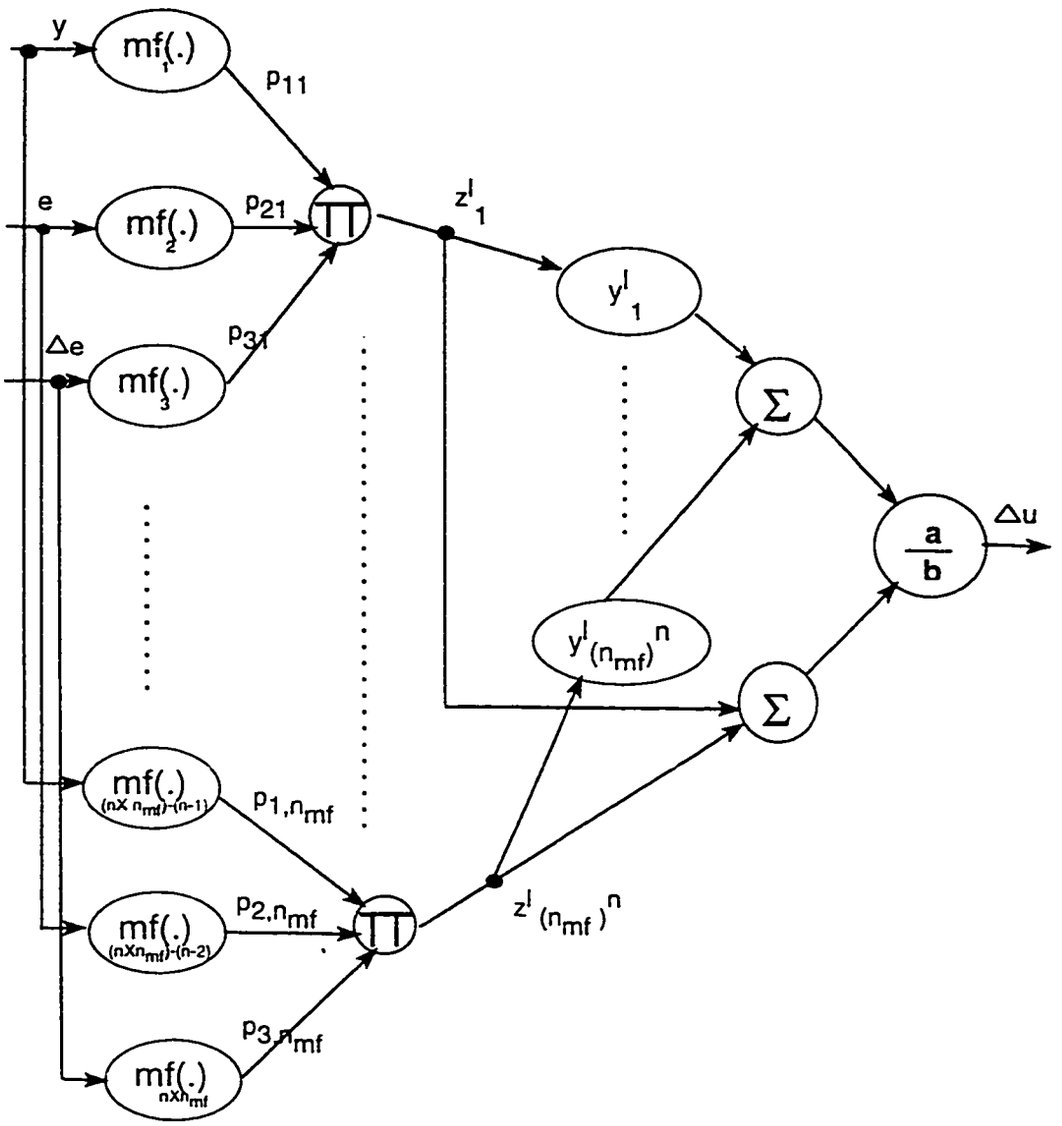


Figure 3.7: Fuzzy Logic controller

dependent on the output. Such dependency can be advantageously used in non-linear plants. In a non-linear plant the control strategy is required to be changed with the operating point. In the proposed strategy the plant output is assumed to convey the operating point information. If this is not the case, the controller should receive an appropriate variable to infer the operating point, instead of the plant output.

It should be noted that Figure 3.7 is a simplified version of the FLC. The original FLC contains two sets of membership functions; the input membership functions and the output membership functions. In the simplified diagram only one parameter of the output membership function, the centroid $y^l$is shown and used. Details of this simplification could be seen in Wang [34].

The fuzzy logic structure contains essentially two layers that contains some tunable functions. While the other layers are the multiplication, addition and division layers. The first layer contains the membership functions. For each input, membership functions (explained in previous chapter) are initially defined by using intuition or expert experience.

The membership functions used are gaussian and their shape is similar to the Bell curve. These are characterized by the centroid, peak and variance as

shown in Figure 3.8. Hence the parameters of the FLC are the centroids of the membership functions μ^l, the variance of the membership functions σ^l and the centroids of the output membership functions y^l. Peaks of membership

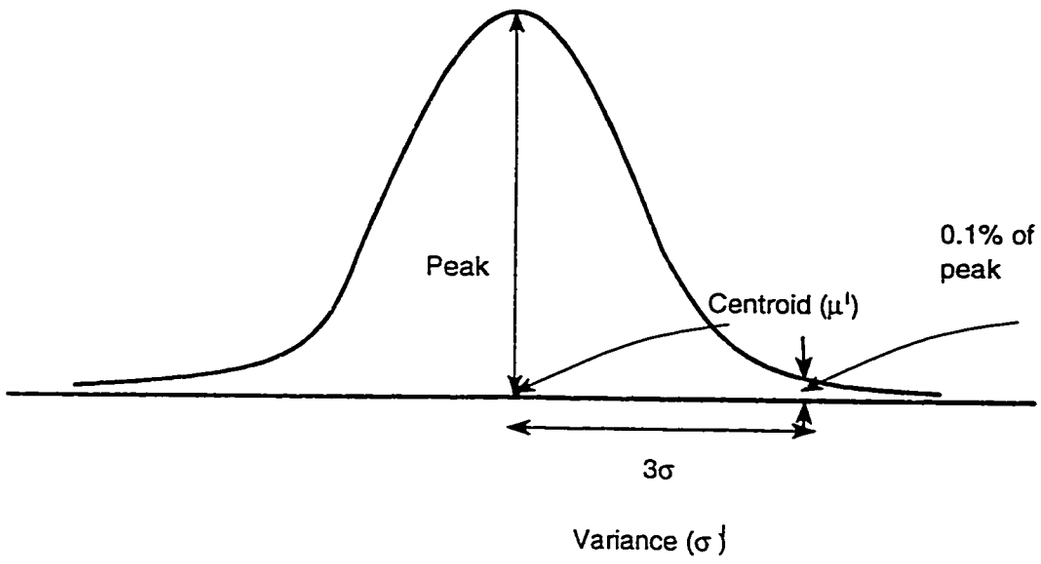


Figure 3.8: Gaussian Membership function

functions could also be tuned but this tuning is essentially the same as that of the output membership function centroids y^l (as shown in Wang [34]). The superscript l is used to distinguish between the fuzzy tunable parameters and the other tunable parameters (i.e superscript l is used for the fuzzy ones). The operation of the FLC could be explained in the following steps

1. The outputs of the membership functions are calculated using the input values of e, Δe and y. Referring to Figure 3.7 a sample calculation can be given as follows:

$$p_{1i} = e^{-(\frac{y-\mu_i^l}{\sigma_i^l})^2} \qquad i = kn+1 \quad for \qquad k = 0,..(n_{mf}-1) \qquad (3.6)$$

In Equation 3.6 first subscript of p shows the input number i.e y, e and Δe respectively represented as 1,2 and 3. The subscript of μ^l and σ^l represents the ith membership function. The number of membership functions are termed as n_{mf} and the number of the inputs to the FLC by n ($n = 3$ for our example).

2. The composite membership functions are obtained by multiplying the membership function of each input. This gives a total of $M = (n_{mf})^n$ composite membership functions. This multiplication is equivalent to *logical AND* operation as shown by Takagi et al. [9] and Wang [34]. For example

$$z_m^l = p_{1i}p_{2j}p_{3k}$$

Where

$$i,j,k = 1..n_{mf} \qquad m = 1..M \qquad M = (n_{mf})^n$$

We define z^l is a vector of $(n_{mf})^n$ values (as shown in the Figure 3.7). Instead of multiplication, taking the minimum of the membership function will also represent the same operation as presented in Takagi [9]. But for training, it cannot be incorporated easily because of its discontinuous nature.

3. z^l vector is multiplied by y^l vector, element by element to produce a vector of length M (Recall that y^l contains the centroid of the output membership function).

4. This is followed by the defuzzifier block where the output of the multipliers z^l values are used to form the weighted average using output centroid weights y^l. y^l is essentially a vector of M values. This will constitute the input needed to control the plant. Hence the input Δu of the plant is given by :

$$\Delta u = \frac{\Sigma_{i=1}^{M} z_i^l y_i^l}{b}$$

Where

$$b = \Sigma_{i=1}^{M} z_i^l$$

With the initial selection of the FLC parameters the output of the controller

most likely will not be able to drive system optimally according to the objective function. Hence training (or tuning) of the parameters is necessary. The training of the FLC could be done based upon a variety of objective functions . Some of these cases are given below:

3.3 Objective Function Selection

Objective functions to be minimized could represent different physical specifications. A general objective criterion that could incorporate a variety of goals to be achieved can be written as follows:

$$J = \frac{1}{2}\Sigma_{t=1}^{k}||C(y_d - y)||^2 + ||Du||^2 \tag{3.7}$$

Where y_d is the desired output, y is the output of the system, u is the input to the system, k is the number of time steps for which the system is simulated, while C and D could be constants or moving average filters of the following form

$$C(q^{-1}) = c_0 + c_1 q^{-1} + ...$$

and

$$D(q^{-1}) = d_0 + d_1 q^{-1} + ...$$

where q^{-1} is the time delay operator. The choice of the constants $c_0, c_1, ...$ and $d_0, d_1, ...$ allows the objective function to account for different aspects of the control system optimization. Some typical selections of C and D polynomials are discussed below:

Case I

Let $C = 1$ and $D = 0$ in the criterion function defined in Eq. 3.7. Then the objective function reduces to:

$$J = \frac{1}{2}\Sigma_{t=1}^{k}||(y_d - y)||^2 \tag{3.8}$$

In this case, the controller design attempts to align the plant output to the desired output, which is usually taken as the output of a judiciously chosen reference model. The gradient of J with respect to a fuzzy parameter w is given by

$$\frac{\partial J}{\partial w} = -\Sigma_{t=1}^{k}(y_d - y)\frac{\partial y}{\partial w} \tag{3.9}$$

Where w is the appropriate weight to be updated and k is the total number of time steps for the simulation of the system. w consists of y^l,μ^l and σ^l, which are the FLC tunable parameters. BPD technique can be used in order to compute $\frac{\partial y}{\partial w}$.

Case II

Let $C = 1$ and $D = d_0 - d_1 q^{-1}$. Substituting these values in Equation 3.7, we get

$$J = \frac{1}{2}\Sigma_{t=1}^{k}||(y_d - y)||^2 + ||(d_0 - d_1 q^{-1})u||^2 \tag{3.10}$$

or

$$J = \frac{1}{2}\Sigma_{t=1}^{k}||(y_d - y)||^2 + ||d_0 u(t) - d_1 u(t-1)||^2 \tag{3.11}$$

The physical significance of the objective function is that the variation in the input is penalized. This can be better understood by letting $d_o = d_1$. The gradient of Equation 3.11 with respect to w is given by:

$$\frac{\partial J}{\partial w} = \Sigma_{t=1}^{k} - (y_d(t) - y(t))\frac{\partial y}{\partial w} + Du(t)D\frac{\partial u}{\partial w} \tag{3.12}$$

or

$$\frac{\partial J}{\partial w} = \Sigma_{t=1}^{k} - (y_d - y)\frac{\partial y}{\partial w} + (d_0 u - d_1 u(k-1))(d_0\frac{\partial u}{\partial w} - d_1\frac{\partial u(k-1)}{\partial w}) \tag{3.13}$$

Where w assumes the appropriate parameter values . In order to compute the above. one has to calculate $\frac{\partial y}{\partial w}$, $\frac{\partial u}{\partial w}$ and $\frac{\partial u(t-1)}{\partial w}$.

Case III

In some cases, specifications are such that more penalty on the input variations is required at steady state as compared to the transient part. This is intuitively clear that in the transient period, input should be relatively free to take any magnitude to drive the plant to the final state while at the steady state for a type zero plant, input variations should be forced to be small so that plant settles to the steady state value. An objective function that will ensure such a behavior of the input could be

$$J = \frac{1}{2}\Sigma_{t=1}^{k}||(y_d - y)||^2 + (Ht)^V||Du||^2 \tag{3.14}$$

Where t is the time, while H and V are constants that are chosen depending on the convergence rate of the parameter values. The gradient with respect to w is

$$\frac{\partial J}{\partial w} = \Sigma_{t=1}^{k} - (y_d - y)\frac{\partial y}{\partial w} + (Ht)^V(Du)\frac{\partial u}{\partial w} \tag{3.15}$$

Equation 3.15 shows that $\frac{\partial y}{\partial w}$ and $\frac{\partial u}{\partial w}$ are to be calculated in order to get $\frac{\partial J}{\partial w}$.

During the training, the objective function may also be changed depending upon the response obtained. For example if the magnitude of input is large, the penalty factor on the input may be increased. Using the above

objective functions, training schemes for the FLC will be formulated for the regulation and servo problems. First we discuss the regulator problem.

3.4 Regulator Problem

The closed loop control system is shown in Figure 3.9. In a regulator problem u_c is zero. Thus e equals $-y$. Hence passing the value of y to FLC through the block k_y is not necessary. Thus for a regulator problem k_y should be zero. Nevertheless we will follow the subsequent derivation with non-zero ky in order to enable easy extention of the results to the servo problem.

The variables k_e,k_{dc},k_u and k_y in the figure are the scaling factors used in order to introduce more degrees of freedom in the control system. The number of adjustable parameters of the controller are increased when scaling factors are used which facilitate the function approximation capacity of the controller. As it is a well known fact that the purpose of the controller is to produce a mapping to modify the dynamics of the plant to achieve the desired response.

In the case of the regulator problem, the output of the system is driven to zero from an initial condition. The initial parameters of the FLC used

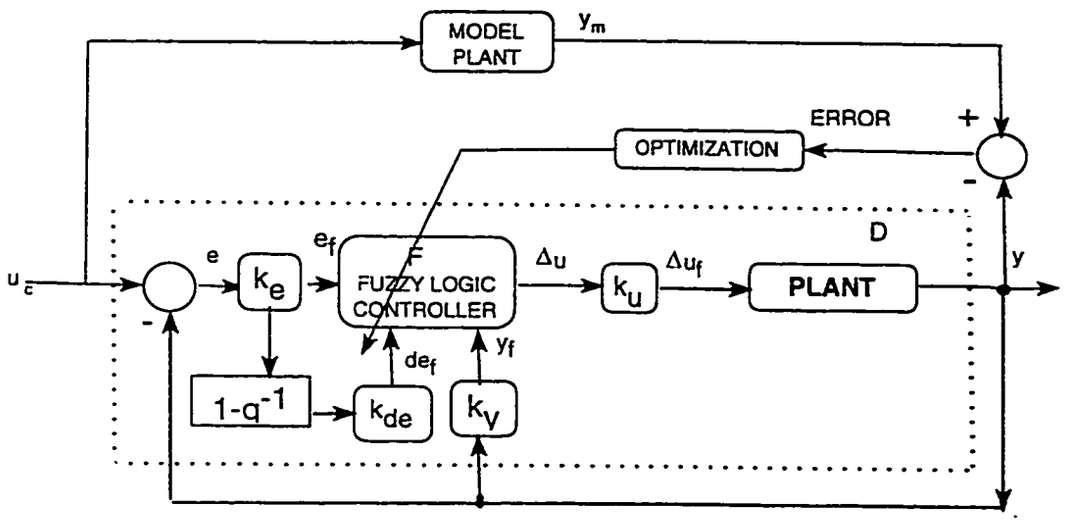


Figure 3.9: Closed Loop control system (Regulator Problem)

in the closed loop system of Figure 3.9 are chosen by operator's experience. This will be elaborated further in a later stage. This FLC is then trained to minimize the chosen objective function. In order to obtain the training equation for the parameters, a linearized equivalent of the control system is drawn as shown in Figure 3.10. The objective function can be taken as that of case I of the previous section as given by Equation 3.8 with $D = 0$.

In order to compute the gradients, the plant is replaced by its input-output linearized model. The blocks which are already linear remain as linear (such as block containing $1 - q^{-1}$) while BPD's for the other blocks are calculated that are used in the place of the non-linear blocks. The parameters which are to be updated are shown as inputs as explained in the Section 3.1.

To keep the diagram simple ∂w is shown as one input. However, it consists of the values ∂y^l,$\partial \mu^l$ and $\partial \sigma^l$ as different fuzzy parameters that are to be tuned. The superscript F shows that these are the BPD's of the FLC. While other partial derivatives correspond to other blocks of the closed loop control system. The subscript f is used to distinguish the variables from their scaled value. For example e_f is the value of e multiplied with its scaling factor k_e. The simplified linearized equivalent is shown in Figure 3.11. Using Mason Gain Formula [48], it could be seen that all the loops are touching. The

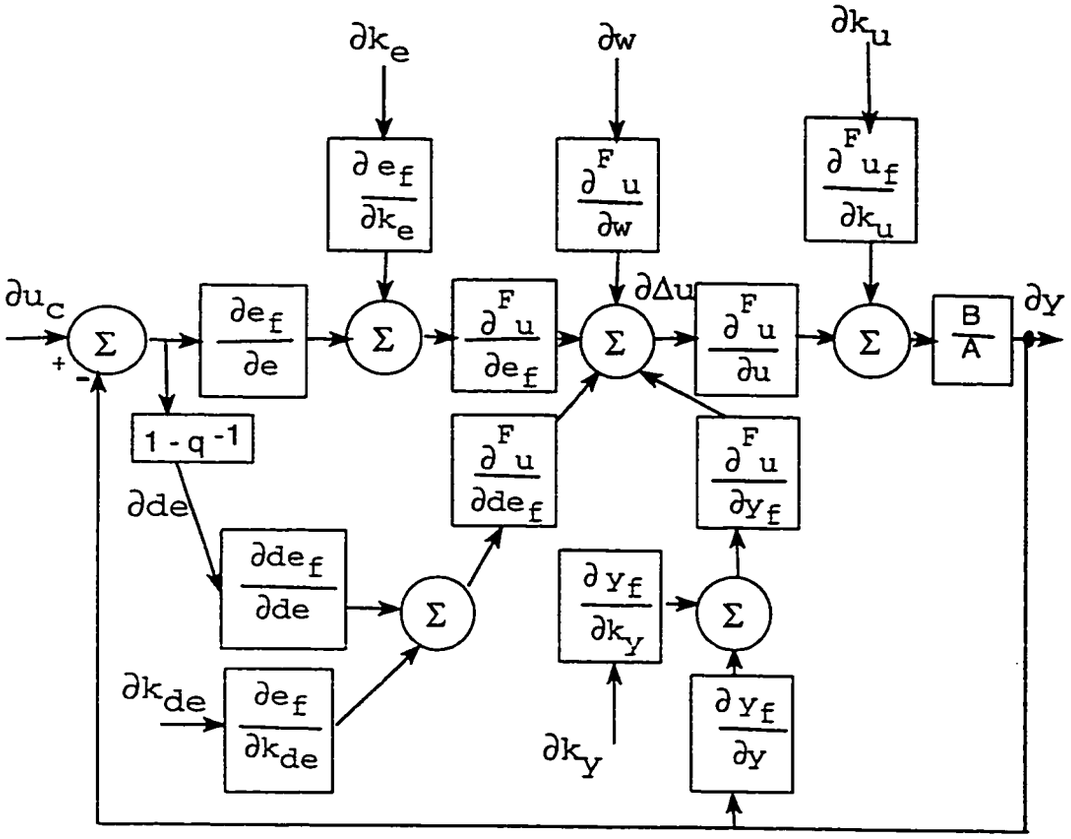


Figure 3.10: linearized equivalent of the closed loop control system

following gradients for the output of the system with respect to the weights of the controller are used to tune the FLC parameters represented by w.

$$\frac{\partial y}{\partial w} = \frac{\partial^F \Delta u}{\partial w} k_u \frac{B}{A(1-\zeta)} \tag{3.16}$$

where w could be y^l, μ^l or σ^l i.e the parameters of the FLC and ζ is given by:

$$\zeta = \frac{(k_e \beta k_u B + (1-q^{-1}) k_{de} \gamma k_u B + k_y \alpha k_u B)}{A(1-q^{-1})} \tag{3.17}$$

Where α , β and γ are defined below using Figure 3.12 that shows the linearized controller. In this figure mf_i' are the gradients of the membership functions computed using the values of their inputs at the previous time instant. mf_i' take the values of derivatives with respect to the parameter w for which $\frac{\partial \Delta u}{\partial w}$ is to be calculated. The exploded view of the linearized equivalent diagram is shown in Figure 3.13. Figure 3.13 shows the sample linearized equivalent for the parameters having index 1, similar diagrams for the whole network of Figure 3.12 enable to derive the required gradients.

The parameters α, β and γ are defined as :

$$\alpha = \frac{\partial^F \Delta u}{\partial y} \tag{3.18}$$

$$\beta = \frac{\partial^F \Delta u}{\partial e} \tag{3.19}$$

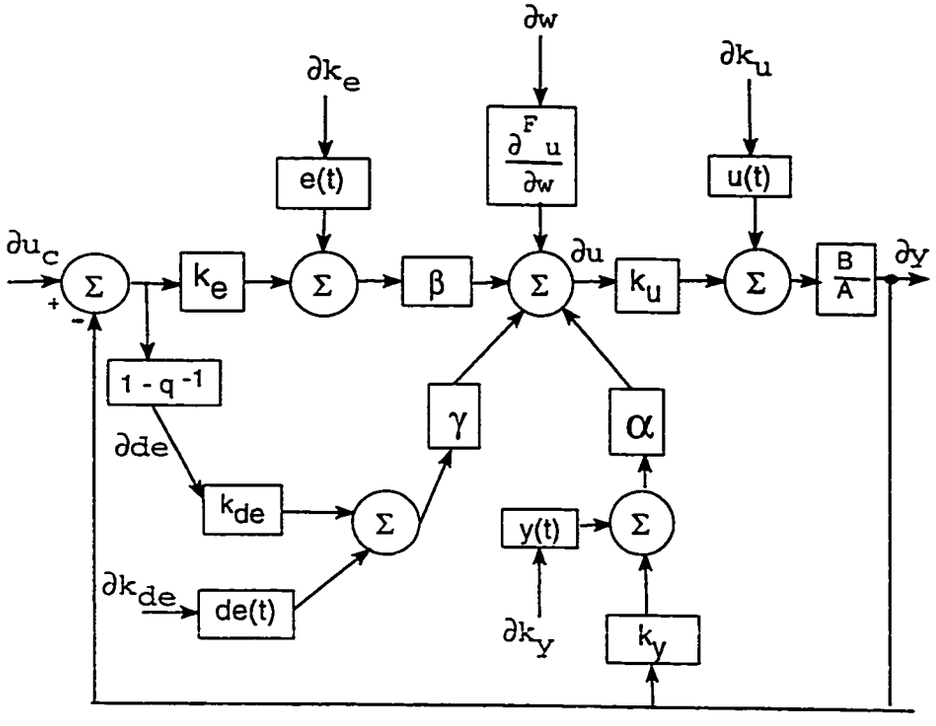


Figure 3.11: Simplified linearized diagram of the closed loop control system

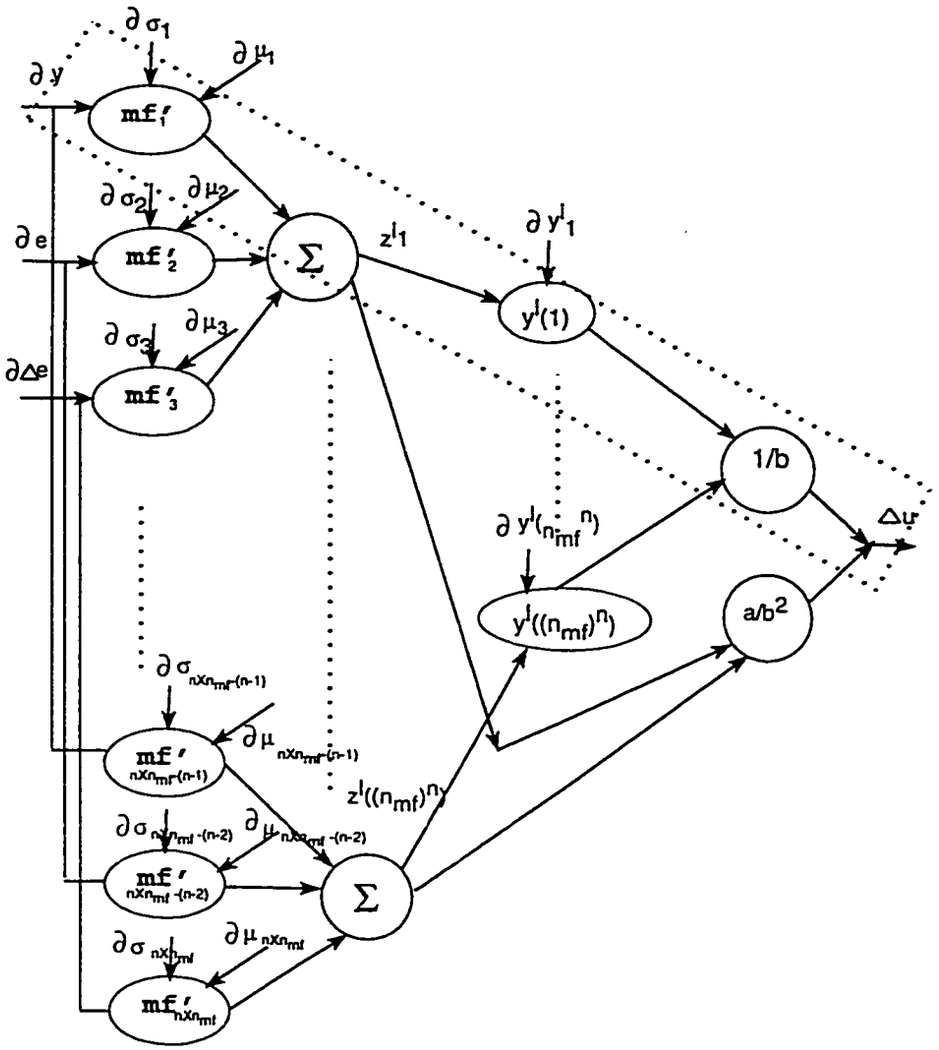


Figure 3.12: Back Propagation through the FLC of fig 3.7 (dotted enclosure is shown in next figure)

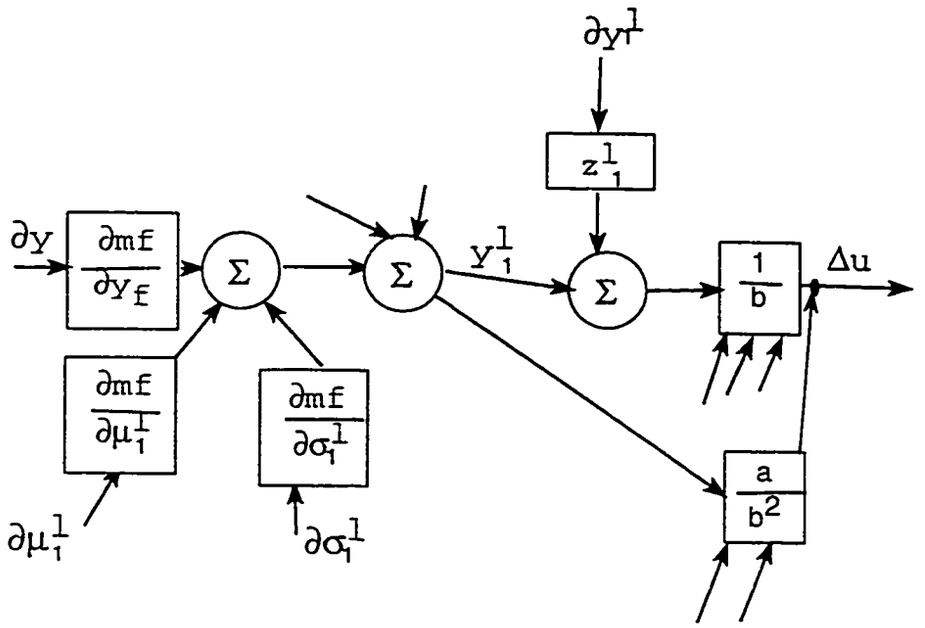


Figure 3.13: Back Propagation through the FLC (exploded view)

$$\gamma = \frac{\partial^F \Delta u}{\partial \Delta e} \tag{3.20}$$

The superscript F shows that the gradient belongs to the block containing the FLC. From Figure 3.12 one may establish

$$(\frac{\partial^F \Delta u}{\partial x})_{ik} = -\frac{(y_i^l - \Delta u) z_i^l (x - \mu_k^l)}{b(\sigma_k^{l\,2})} \tag{3.21}$$

Where x may take any of the values y, Δe or e for the derivation of the respective gradients. i is the index of z^l and y^l on the paths from x to Δu as shown in Figure 3.12. Similarly k is the index of μ^l and σ^l on the path from x to Δu. α, β and γ are the summation of gradients over all $i's$ that are in the path from the respective input to Δu. For the computation of Equation 3.16, $\frac{\partial^F \Delta u}{\partial w}$ are calculated using Figure 3.12. As explained earlier that w contains the parameters of the FLC that could be the output centroids weights, variance of the membership functions or the centroid of the output membership functions. From the backpropagation network of the controller in Figure 3.12 the derivatives could be calculated. Gradients of the FLC can also be obtained using the Ordinary Partial Derivatives as shown in Wang [34].

For y^l, μ^l and σ^l, the gradients as derived in Wang [34] are

$$\frac{\partial^F \Delta u}{\partial y_i^l} = \frac{z_i^l}{b} \qquad i = 1..M \tag{3.22}$$

$$\frac{\partial^F \Delta u}{\partial \mu_i^l} = \frac{(y_k^l - \Delta u) z_k^l (x - \mu_i^l)}{b(\sigma_i^{l^2})} \tag{3.23}$$

$$\frac{\partial^F \Delta u}{\partial \sigma_i^l} = \frac{(y_k^l - \Delta u) z^l (x - \mu_i^l)^2}{b(\sigma_i^{l^3})} \tag{3.24}$$

Where x, k and i have the same role as explained above. z^l, M and b are defined in the section 3.2.1. Using the equations 3.16, 3.22, 3.23 and 3.24 we get the gradients of the output of the system with respect to parameters y^l, μ^l and σ^l as follows:

$$\frac{\partial y}{\partial y_i^l} = \frac{k_u(z_i{}^l) B}{(1-\zeta) A b} \qquad i = 1..M$$

$$\frac{\partial y}{\partial \mu_i^l} = \frac{2(y_k^l - \Delta u) z_k^l (x - \mu_i^l) k u B}{b \sigma_i^2 (1-\zeta) A} \qquad i = 1..n * n_{mf}$$

$$\frac{\partial y}{\partial \sigma_i^l} = \frac{2(y_k^l - \Delta u) z_k^l (x - \mu_i^l)^2 k u B}{\sigma_i^3 (1-\zeta) A} \qquad i = 1..n * nmf$$

Where x, k and i have the same role as explained above. As shown in the the Section 3.3, when $D \neq 0$ we need the gradients for input also, which are computed as follows

$$\frac{\partial \Delta u}{\partial y_i^l} = \frac{z_i^l}{(1-\zeta) b} \qquad i = 1...M$$

$$\frac{\partial \Delta u}{\partial \mu_i^l} = \frac{2(y_k^l - \Delta u) z_k^l (x - \mu_i^l)}{b\sigma_i^2(1-\zeta))} \qquad i = 1..n * nmf$$

$$\frac{\partial \Delta u}{\partial \sigma_i^l} = \frac{2(y_k^l - \Delta u) z_k^l (x - \mu_i^l)^2}{\sigma_i^3(1-\zeta)} \qquad i = 1..n * nmf$$

Where x, k and i have the same role as explained above. To increase the capability of the controller of approximating non-linear functions we can also tune up the scaling factors. For tuning of the scaling factors using Mason Gain Formula [48] we get

$$\frac{\partial y}{\partial k_e} = \frac{e(k)\beta k_u B}{A(1-\zeta)}$$

$$\frac{\partial y}{\partial k_u} = \frac{\Delta u(k) B}{A(1-\zeta)}$$

$$\frac{\partial y}{\partial k_{de}} = \frac{de(k)\gamma k_u B}{A(1-\zeta)}$$

$$\frac{\partial y}{\partial k_y} = \frac{y(k)\alpha k_u B}{A(1-\zeta)}$$

where k is the time step. $e(k)$, $de(k)$ and $y(k)$ are the values at the k^{th} time instant. We can also use a *bias* to increase the degrees of freedom. The *bias* is normally used in the neural networks. It is a link having a specified weight connected to a constant input. It is especially required for tuning the steady state response. Addition of a *bias* is shown in Figure 3.14.

This bias is introduced essentially to help in fine tuning the response.

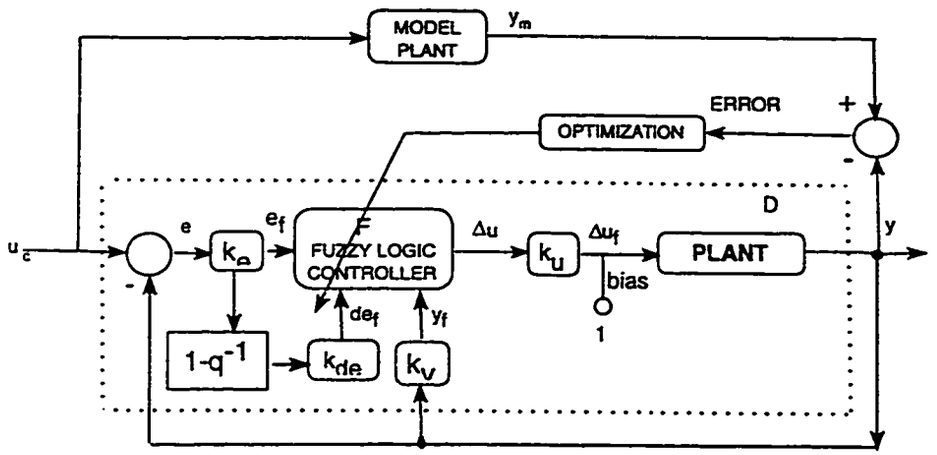


Figure 3.14: Introduction of bias in the controller

The gradient for the training of the bias is as follows

$$\frac{\partial y}{\partial bias} = \frac{\beta k_u k_e B}{A(1-\zeta)}$$

The training method using the above equations will be described in detail in Section 3.7 . Next we discuss the servo problem.

3.5 Servo Problem

In the case of servo problem, the plant is driven by the controller to follow a reference command. We will consider only the step like command. In our context the difference between the regulator and the servo problem is that in the latter case an integrator is used in the forward path. The control system

configuration is shown in Figure 3.15. Before deriving the equations needed

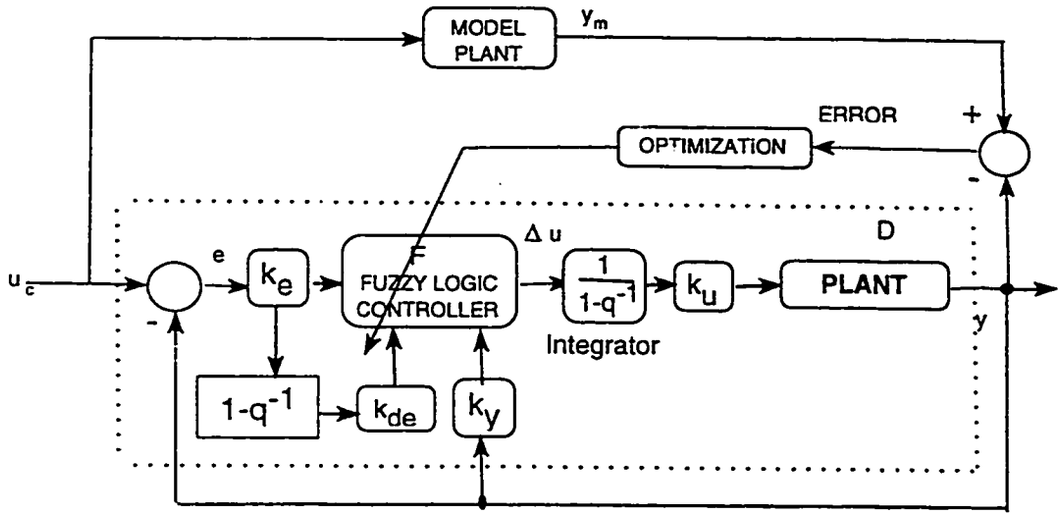


Figure 3.15: Closed Loop control system with Integrator

for training. a result is presented below concerning the use of integrators in the forward loop for the control of non-linear plants.

3.5.1 Use of Integrator in control of non-linear plant

In the case of linear plants, we can use integrator for the tracking of a step command. It ensures zero steady state error in case of type zero plants as could be seen in Dorf [48]. The proof exists for linear plants but no such result is available for the non-linear plants. An effort toward this end is

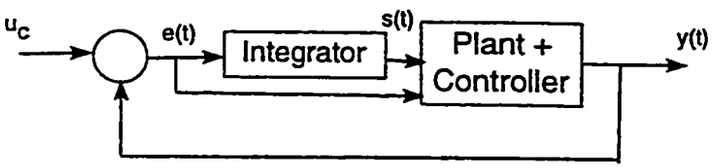


Figure 3.16: Integrator before the plant

presented below; Consider the control configuration in Fig. 3.16 where $e(t)$ is the error, $s(t)$ is the integrated value of the error and $y(t)$ is the output of the plant, $y(t)$ is the function of $e(t)$, $s(t)$ and their delayed values and the delayed values of $y(t)$. Let the controller plant combination be described by the function $\psi(.)$ such that $y(t)$ is represented as

$$y(t) = \psi(e(t-1), e(t-2)...., s(t-1), s(t-2)...., y(t-1), ...)$$

For simplicity, the controller plant delay is assumed to be unity.

Proposition : if $\psi(t)$ is monotonic in $s(t-1)$ for $y(t) \in \mathbf{A}$ $\forall t$; $\mathbf{A}$ being subset of $\mathbf{R}$, then $\forall u_c \in \mathbf{A}$ the steady state value of $y(t)$ is defined as:

$$y_{ss}(t) = Lim_{t \to \infty} y(t)$$

would either

(a) converges to u_c

(b) oscillates indefinitely

or (c) leaves **A**.

The proof of the proposition can be found in Ahmed [47].

3.5.2 Training Equations for the Servo problem

In this case the training equations are slightly different as compared to the regulator case. The reason being the incorporation of the integrator. The linearized equivalent of the closed loop system is shown in Figure 3.17. Using Figure 3.17 we get the following training equations

$$\frac{\partial y}{\partial y_i^l} = \frac{k_u(z^l{}_i)B}{(1-q^{-1})(1-\zeta)Ab} \qquad i = 1..M$$

$$\frac{\partial y}{\partial \mu_i^l} = \frac{2(y_k^l - \Delta u)z_k^l(x-\mu_i^l)k_u B}{b\sigma_i^2(1-\zeta)(1-q^{-1})(1-q^{-1})A} \qquad i = 1..n*nmf$$

$$\frac{\partial y}{\partial \sigma_i^l} = \frac{2(y_k^l - \Delta u)z_k^l(x-\mu_i^l)^2 k_u B}{\sigma_i^3(1-\zeta)(1-q^{-1})A} \qquad i = 1..n*nmf$$

Where x, k and i have the same role as explained previously. As shown in the previous section, when $D \neq 0$ we also need the gradients of the input with respect to the parameters which are obtained as follows:

$$\frac{\partial u}{\partial y_i^l} = \frac{z_i^l}{(1-q^{-1})(1-\zeta)b} \qquad i = 1..M$$

$$\frac{\partial u}{\partial \mu_i^l} = \frac{2(y_k^l - \Delta u)z_k^l(x-\mu_i^l)}{b\sigma_i^2(1-\zeta)(1-q^{-1})(1-q^{-1})} \qquad i = 1..n*nmf$$

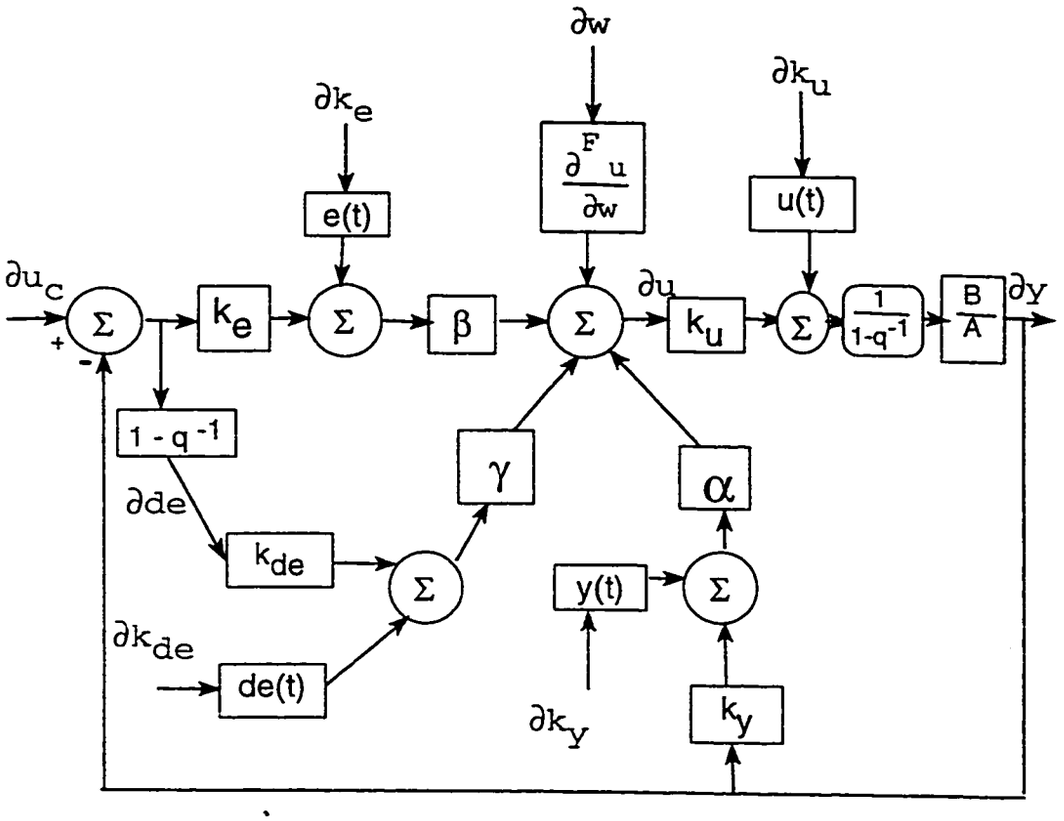


Figure 3.17: Modified Linearized diagram using the integrator

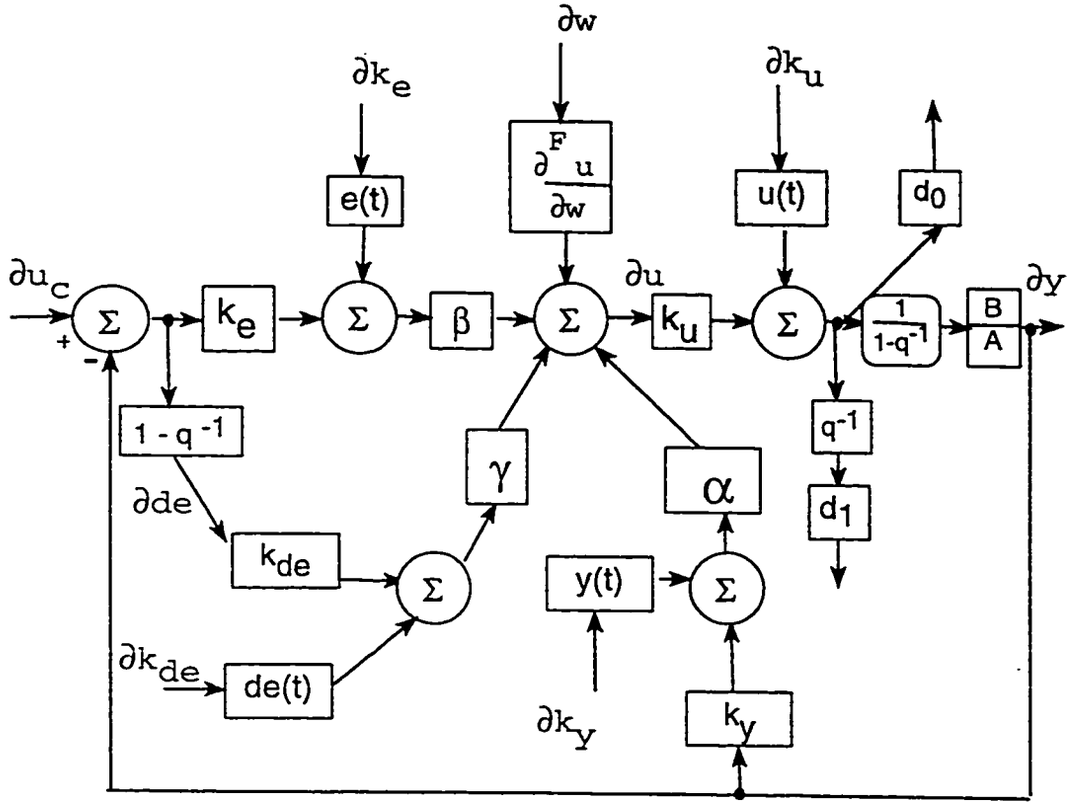


Figure 3.18: Modified Linearized diagram using the integrator (with input penalty)

$$\frac{\partial u}{\partial \sigma_i^l} = \frac{2(y_k^l - \Delta u) z_k^l (x - \mu_i^l)^2}{\sigma_i^3 (1-\zeta)(1-q^{-1})} \qquad i = 1..n * nmf$$

Where x takes the value y, Δe and e for the derivation of the respective gradients. k is the index of z^l and y^l on the paths from x to Δu in the Fig. 3.12. Similarly i is the index of μ^l and σ^l on the path from x to Δu.

The above equations are consistent with the definition of z^l, b and M as shown in the previous section. After the presentation of the training equations some stability issues for the scheme are touched below.

3.6 Stability Issues

In the proposed training scheme stability could not be guaranteed in absolute terms. But this is not a noticeable drawback herein because the training technique is purely off-line. Prior to the implementation, the network structures, parameters and the learning rates can be altered through trial and error, or based on heuristics until a satisfactory response is obtained. Hence initially the controller will be trained for scenarios that could arise in the normal operation of the plant and then it will be hard-wired for the implementation. During the training, all possible setpoints in the normal operating range of the system should be included. Once the controller is trained for all oper-

ating regions of the plant, it is expected to perform satisfactorily during the implementation stage.

3.6.1 Stability proof near the minimum

Although no general stability proof exists for the training algorithm, in the following an attempt is made to prove the stability of the gradient computation in the vicinity of optimal parameters . It is based on similar derivation for neural networks presented in Ahmed [47].

Using the locally linearized network formed by BPD's as shown in Figure 3.18, the linearized input output relationship is given by

$$\overline{y}(t) = G_c(q)\overline{u_c}(t)$$

Here without loss of generality, it is assumed

$$k_e = k_{de} = k_y = k_u = 1$$

for the sake of simplicity. Where $G_c(q)$ can be obtained from the linearized equivalent diagram of Figure 3.11 as

$$G_c(q) = \frac{\beta B}{A(1-q^{-1}) + \beta B + \gamma(1-q^{-1})B - \alpha B} = \frac{\beta B}{A_c} \qquad (3.25)$$

$\overline{y}$ and $\overline{u_c}$ are the incremental values of $y(t)$ and $u_c(t)$ and

$$A_c = A(1-q^{-1}) + \beta B + \gamma(1-q^{-1})B - \alpha B$$

The parameters α, β and γ are defined in Equations 3.18, 3.19 and 3.20. As soon as the closed loop system approaches the reference model, the input-output relationship is given by

$$G_m(q) = \frac{q^{-d}B_m(q)}{A_m(q)} \tag{3.26}$$

$B_m(q)$and $A_m(q)$ are the parameters of linear reference model while d is the time delay. Near the optimal point we could use the equations as presented in Astrom [53].

$$A_0 A_m = A_c \qquad and \qquad q^{-d}A_o B_m = \beta B \tag{3.27}$$

where A_o is an observer polynomial. If the closed loop system is stable it will also be Schur. A_c is the denominator of the transfer function in Eq. 3.25. It can be seen from Eq. 3.27 that A_c is Schur.

From Figure 3.12 of linearized network the sensitivity derivative (i.e the gradient with respect to the adjustable parameters) is obtained as

$$\Gamma(t) = G_w(q)[\frac{\partial^F u}{\partial w}^T]$$

where the value of $G_w(q)$ is given by

$$G_w(q) = \frac{B}{(A(1-q^{-1}) + \beta B + \gamma(1-q^{-1})B - \alpha B)} \tag{3.28}$$

or

$$G_w(q) = \frac{B}{A_c} \tag{3.29}$$

Since A_c is schur, Eq. 3.29 implies that

$$\Gamma(t) = G_w(q)[\frac{\partial^F w}{\partial w}^T]$$

is bounded.

3.6.2 Learning Rate and Stability

The training algorithm is based on steepest descent method for which there is no proof that it will converge to the global minimum. Convergence to a suitable minimum depends on the starting parameter values and learning rate. Learning rate could be constant or adaptive. Ahmed [47] presented an adaptive learning rate that can be shown to preserve stability of the training near the optimal parameters as long as $\Gamma(t)$ is bounded.

Usually learning rate is determined by trial and error procedure. Too large learning rate may make the training process unstable. Also it could

miss the global minimum. It can be observed from Eq 3.1 that a large value of η will shift the parameter values to a large degree. Similarly too small learning rate may result in increased computation. A suitable value of η could be figured out by repeated trials.

The convergence in steepest descent method also depends on the starting point. In our simulation study although we started from an initial working controller obtained through trial and error. The initial FLC could be designed in many ways. There exists various parameter sets that may make such a working controller. It may happen that the starting controller is already near a local minimum and it may not improve further. Thus training with various working controllers may be needed to get a satisfactory trained controller.

It is to be noted that the training method is similar to that of MFNN (Multiple feedforward Neural Networks). However, in the case of MFNN one is concerned with the training of its weights. No physical significance could be attributed to these weights. In the case of FLC, we are confronted with different aspects of the layers. In the first layer parameters to be tuned are the variance and centroid of the input membership function while in the third layer centroids of the output membership functions are trained. It is found from simulation study that a small change in the latter is much more

significant as compared to that in former ones. This observation justifies the use of different learning rates for different parameters.

3.7 Computational Issues

No hard and fast rules can be specified for the training of FLC. Due to the nature of the algorithm, training requires heuristics. Some of the heuristics established from intensive experimentation are discussed below:

The training method primarily is user guided, analogous to the neural controller training presented in Ahmed [47]. Initially a working controller is achieved by setting up the Fuzzy IF-THEN rules. These rules could be initialized and modified by observing the response of the system under different operating conditions. By playing with the parameters it is almost always possible to get a stable closed loop control system for at least one setpoint. If this is not possible, other alternatives as explained in the following can be attempted. The complete training process can be described through two phases.

3.7.1 Phase I

- The parameters of the FLC can be initialized randomly. Then the training algorithm is used to update the FLC parameters. It may attain some working control in a number of iterations. This approach may be repeated a few times until a satisfactory response is obtained. In case of random weights, the output centroids should be smaller to start with. From the simulations conducted it is inferred that these parameters change faster. For a randomly initialized FLC most of the times the system will be unstable. Therefore, it is essential that the initial simulation time is small such that the system has bounded output. As soon as a satisfactory output is obtained, the simulation time can be increased gradually

- When the above approach fails to work, the FLC could be started with parameters initialized with human logic and experience. Then, starting from a small simulation time, the training algorithm may be applied to improve these parameters. This simulation time can be gradually increased as the training proceeds.

- The third approach is to acquire a working controller for the given

plant using the standard PID or LQR techniques. FLC is then trained to imitate the input output data of that controller. This can be done by the standard BP training as presented in Rumelhart[49]. This will provide an initial FLC so that it can be fine tuned afterwards. If the degree of non-linearity in the system is high this approach might not work.

3.7.2 Phase II

After the initialization phase one has a stable system that is to be improved so that output of the system minimizes the criterion function. In the user guided training the trained control system is closely monitored. Otherwise the system may become unstable during the training. The response of the system is monitored alongwith the objective function behavior. The learning rates for each of the parameter are adjusted accordingly.

The parameters of FLC do not uniformly effect the system's behavior and generally require to be updated using different learning rates. To obtain convergence, vigilance may be necessary to counter for the different behavior of the FLC parameters. At times, the convergence may be obtained only by

freezing some of the parameters.

One aspect of the FLC training is that the physical significance of the parameters is better known in FLC, as opposed to neural network weights. Neural networks suffer from the disadvantage that no real meaning could be associated with each layer. However in the case of FLC it may be possible to adjust the proper parameters by monitoring the closed loop response during the training. For example, these adjustable parameters could be the centroids when the response exhibits steady state error. Also the narrowing of the membership functions i.e decreasing the variance may help to reduce the error range. At times, the centroids of output membership functions could be adjusted to fine tune the response.

Once the control for one setpoint is obtained, it is time to train for a different setpoint. The parameters achieved from the previous training are preserved and training for the next setpoint is started from these parameter values. It could be difficult to train the controller in the presence of a large transition in the reference command. In this situation the initial training should be conducted with a small transition in the setpoint.

During the training, the setpoint transition may be gradually increased. Another problem that may occur is the overtraining of some parameters.

This means that the training for an earlier setpoint transition is obliterated while the response due to new setpoint transition performs better. This conflict could be resolved by increasing the FLC size. This can be done by increasing the number of membership functions etc.

A further consideration is to identify the proper parameters which are to be tuned to improve the control performance for a particular setpoint transition. The other parameters that are not related to the new setpoint should be frozen. For example, the parameters relating to the membership functions that are not activated by the error values of the new setpoint can be held fixed. For this purpose corresponding elements of leaning rate matrix may be set to zero.

In the presence of an steady state error in the output, during the training a constant *bias* could be added to the FLC (as explained earlier). This is purely a neural network concept. It is trained at the very end as being related linearly to Δu, it dominates the training, causing very slow update of other parameters.

Chapter 4

Simulation Results

To validate the proposed scheme simulation studies are conducted on three examples. These examples represent practical engineering systems.

4.1 A Laboratory Scaled Liquid Level System

A second order model identified from a laboratory scaled Liquid Level system by Billings *et al.* [46] represented by the following NARX model, is considered

$$y(t) = 0.9722y(t-1) + 0.3578u(t-1) - 0.1295u(t-2) - 0.3103y(t-1)u(t-1)$$

$$-0.04228y^2(t-2) + 0.1663y(t-2)u(t-2) - 0.03259y^2(t-1)y(t-2)$$

$$-0.3513y^2(t-1)u(t-2) + 0.3084y(t-1)y(t-2)u(t-2)$$

$$+ 0.1087y(t-2)u(t-1)u(t-2) \tag{4.1}$$

Where $y(t)$ is the output and $u(t)$ is the input. The reference model used to train the closed loop controller is given by

$$y_m(t) - 0.8y_m(t-1) = 0.2u_c(t-1) \tag{4.2}$$

The objective function for training is chosen to be

$$J = \frac{1}{2}\Sigma||y_m - y||^2 \tag{4.3}$$

The negative and positive step responses of the system using an initial FLC that was developed by trial and error. are shown in Figures 4.1 and 4.3, respectively.

It could be seen that for the positive step, the output of the plant can follow the command. However, for a negative step, the plant is oscillatory and completely failed to follow the command. Beside repeated trials it was not

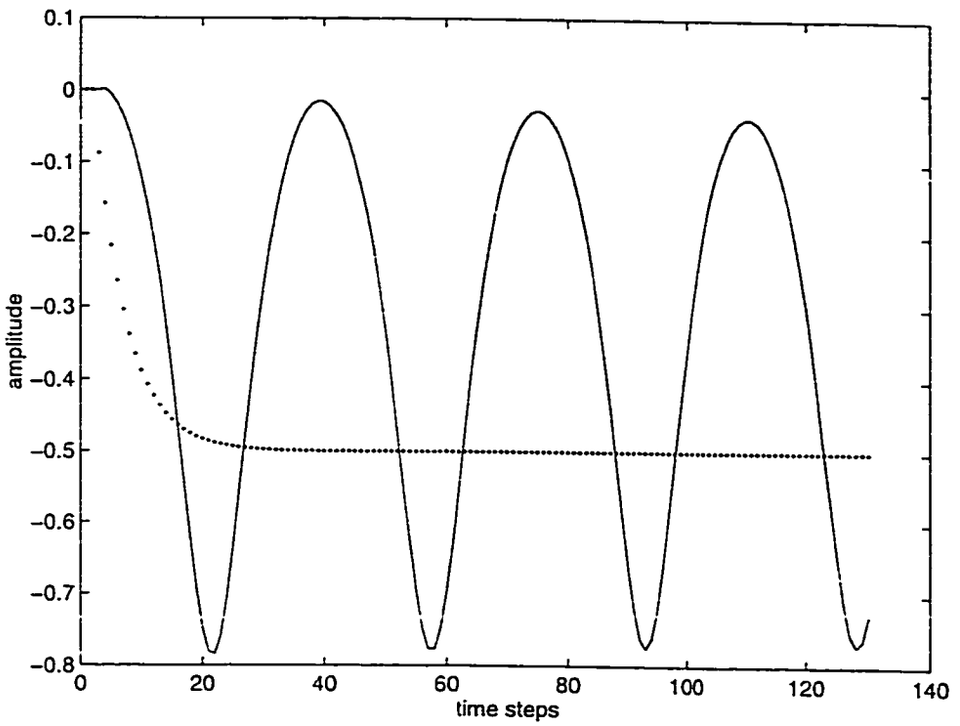


Figure 4.1: Negative step response of the Liquid Level System prior to training (-) and output of reference model (..)

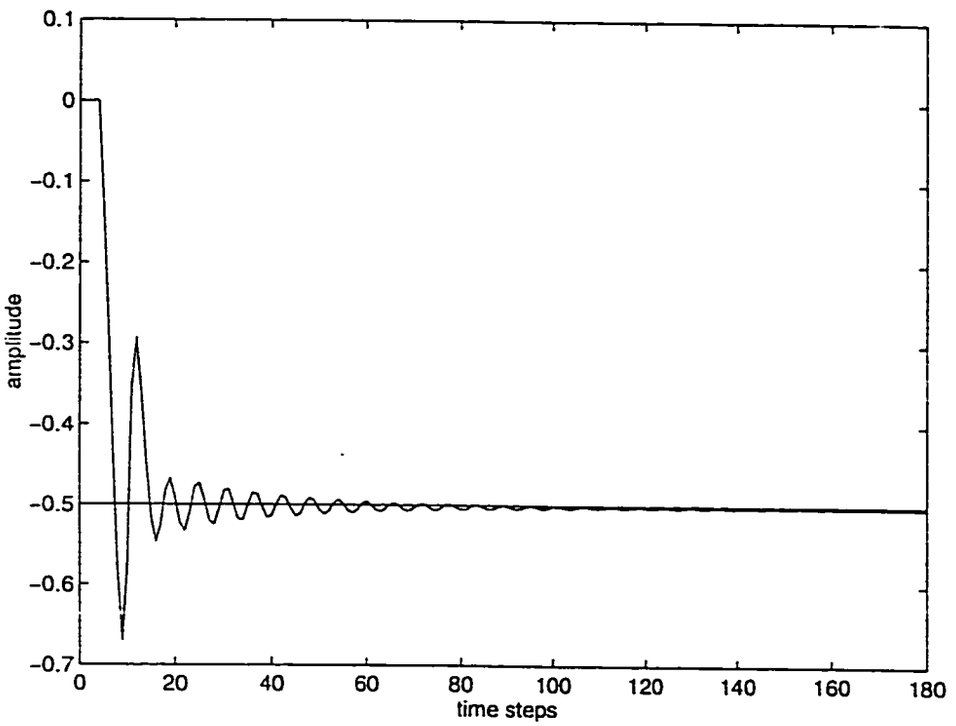


Figure 4.2: Negative step response of the Liquid Level System using the trained FLC

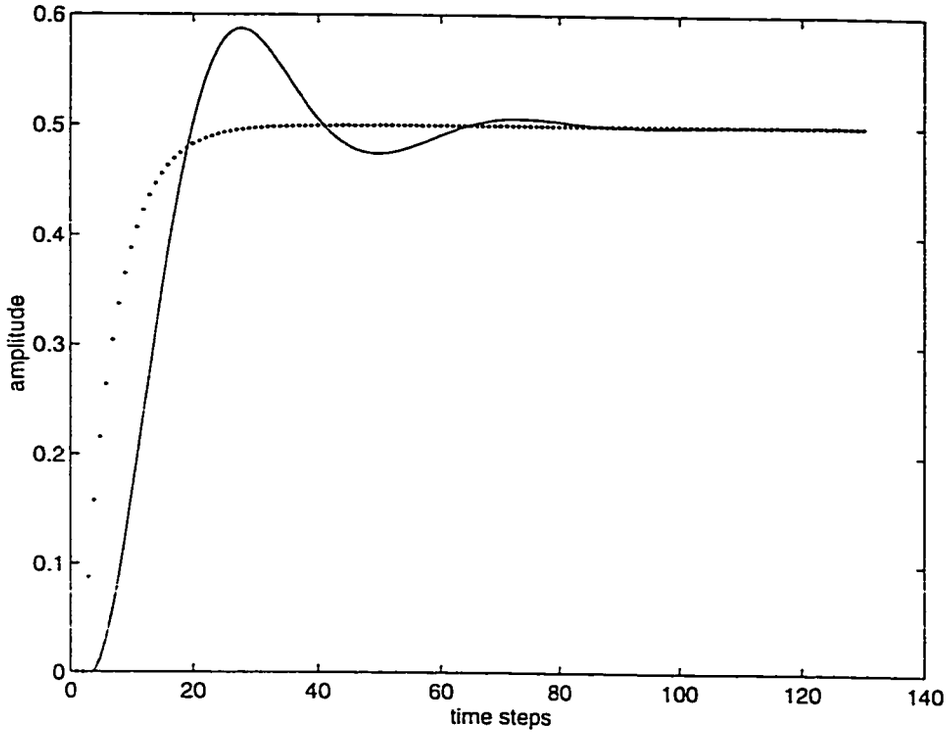


Figure 4.3: Positive step response of the Liquid Level System (-) prior to training and output of reference model (..)

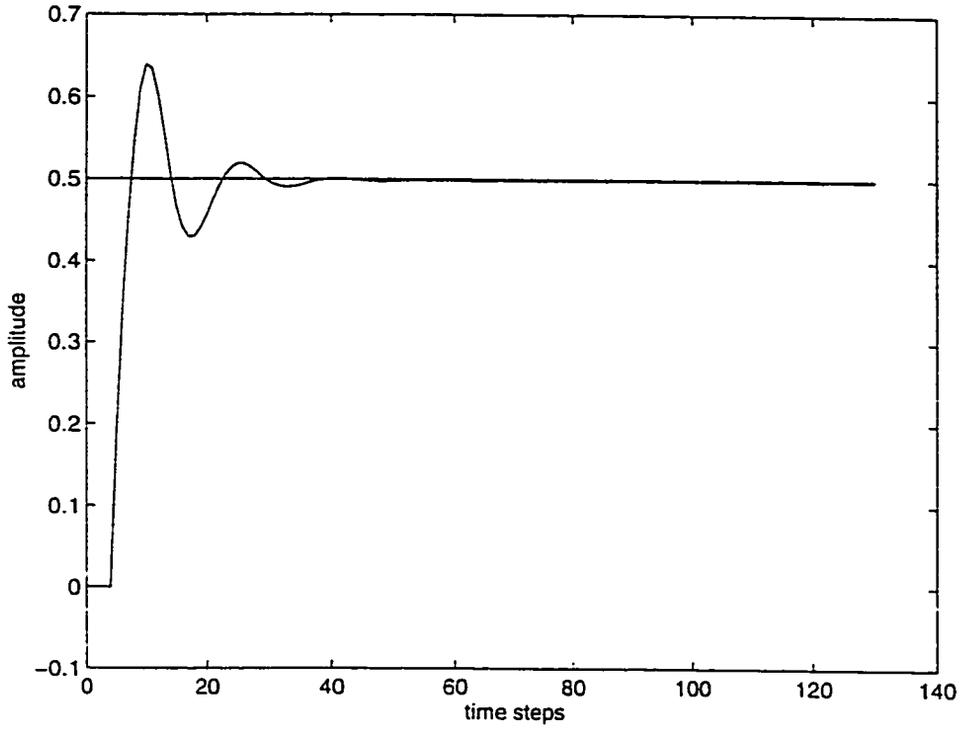


Figure 4.4: Positive step response of the Liquid Level System using the trained FLC

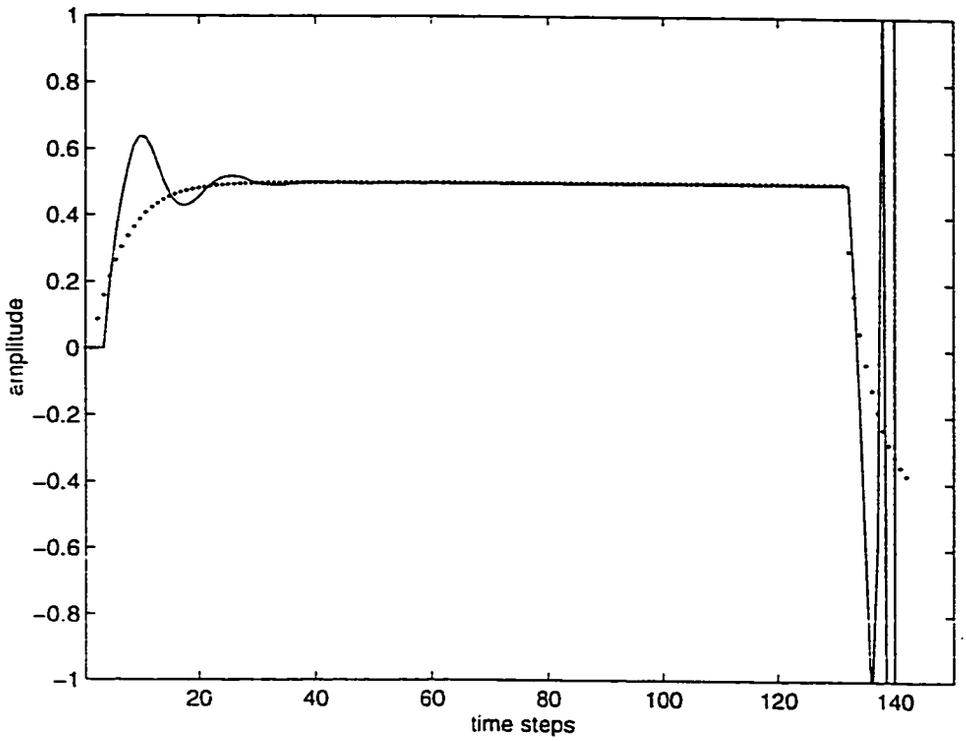


Figure 4.5: Output of the liquid level system for multiple step commands (–) and reference model output(..) using the untrained FLC

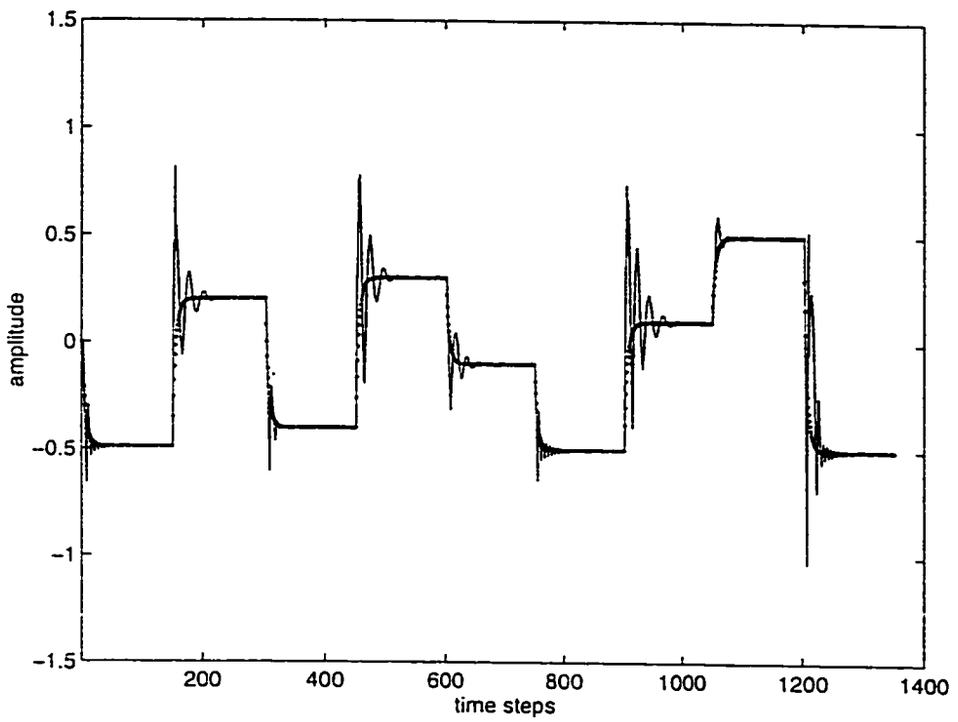


Figure 4.6: Output of liquid level system using trained FLC for random step commands ( ) and reference model output(..)

possible to get a better FLC for the negative step. For the same step inputs the response of the system after training of the FLC is shown in Figures 4.2 and 4.4 respectively.

For multiple step change at the command, the system response due to the untrained FLC is shown in Figure 4.5. Note that the second leg of the response is unstable. With the trained FLC, the output of the system for random setpoint change is shown in Figure 4.6. It could be seen that the output of the system followed the command input satisfactorily.

4.1.1 Robustness of the FLC

In order to assess the robustness of the FLC, simulation studies have been conducted to check the sensitivity of the control system. Tests have been conducted to determine how much variation in the nominal parameter values can keep the system stable. The range of each parameter for which the system remained stable, when the other parameters are held at their nominal values, is shown in the Table 4.1. Figure 4.7 and 4.8 show the step response of the closed loop system with the parameter variations.

Except for one or two parameters the designed controller has been found

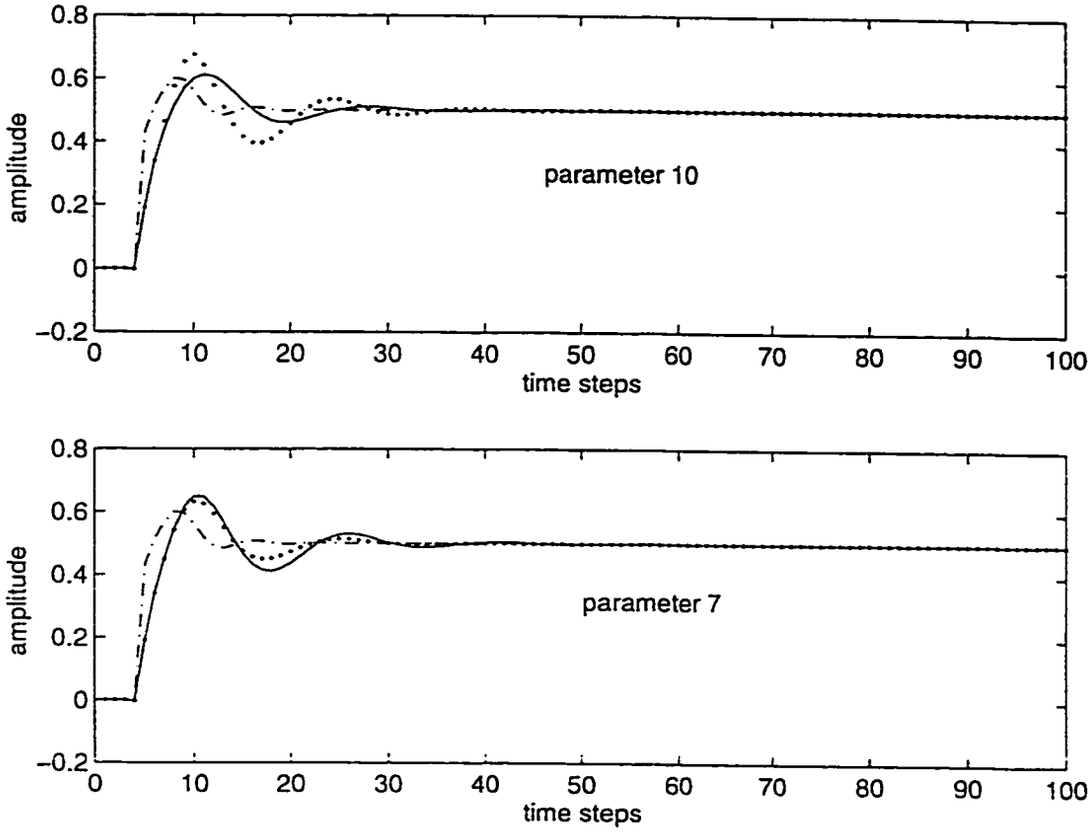


Figure 4.7: Effect of parameter variation on step response of the Liquid Level system Nominal (.-) (parameter 10, -.1087)(parameter 7, 0.03259), Positive variation (..) (parameter 10, -.2174)(parameter 7, 0.06518), Negative variation (..) (parameter 10, -.01087)(parameter 7, .00162)

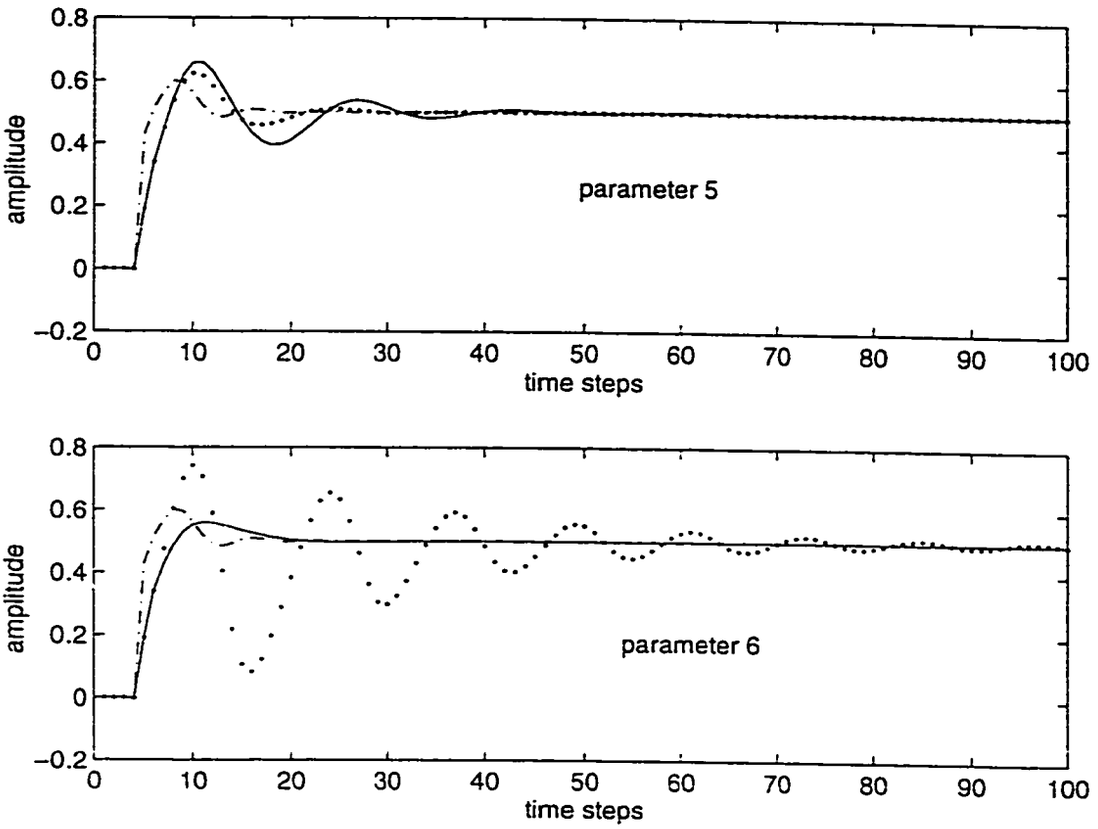


Figure 4.8: Effect of parameter variation on step response of the Liquid Level system Nominal (.-) (parameter 5, -.04228)(parameter 6, -.1663), Positive variation (..) (parameter 5, -.08456)(parameter 6, -0.3326), Negative variation (..) (parameter 5, -.00214)(parameter 6, -.00083)

Parameter No.	Value	+ve variation %	-ve variation %
1	.9722	10	75
2	−.3578	50	65
3	−.1295	100	95
4	−.3203	100	95
5	−.04228	100	95
6	−.1663	100	95
7	0.03259	100	95
8	−.3543	100	75
9	−.3048	100	95
10	−.1087	100	95

Table 4.1: Variations in parameters of liquid level system for which the control system followed the command

to be relatively robust against parameter variations. Although the performance deteriorated due to parameter variations, the control system remained stable for a large range of parameter values.

4.2 Hammerstein Non Minimum Phase plant

Hammerstein plants are the class of non-linear plants in which a plant is modeled by cascading a static non-linearity with a dynamic linear component. A typical Hammerstein plant is shown in the Figure 4.9.

The linear dynamic part chosen for simulation has been non-minimum

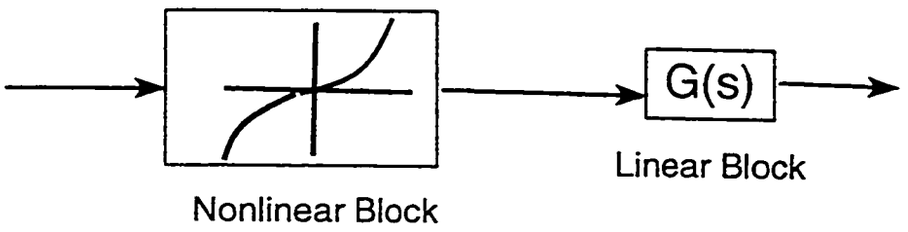


Figure 4.9: Hammerstein Plant

phase in nature. In the case of non minimum phase plants, reference model based controller design does not work. In a model reference based design a component of the controller attempts to assume the inverse plant dynamics in order to cancel the original plant dynamics. But the inverse of the non minimum phase plant is unstable. A controller having such unstable dynamics will produce unbounded control signal.

To overcome this problem the objective function can be chosen to include penalty on the control input amplitude or its variations, alongwith the error. By properly choosing this penalty factor, it may be possible to come up with a controller to produce bounded control input. This controller will enable the system to closely follow the reference model. Consider the following modified

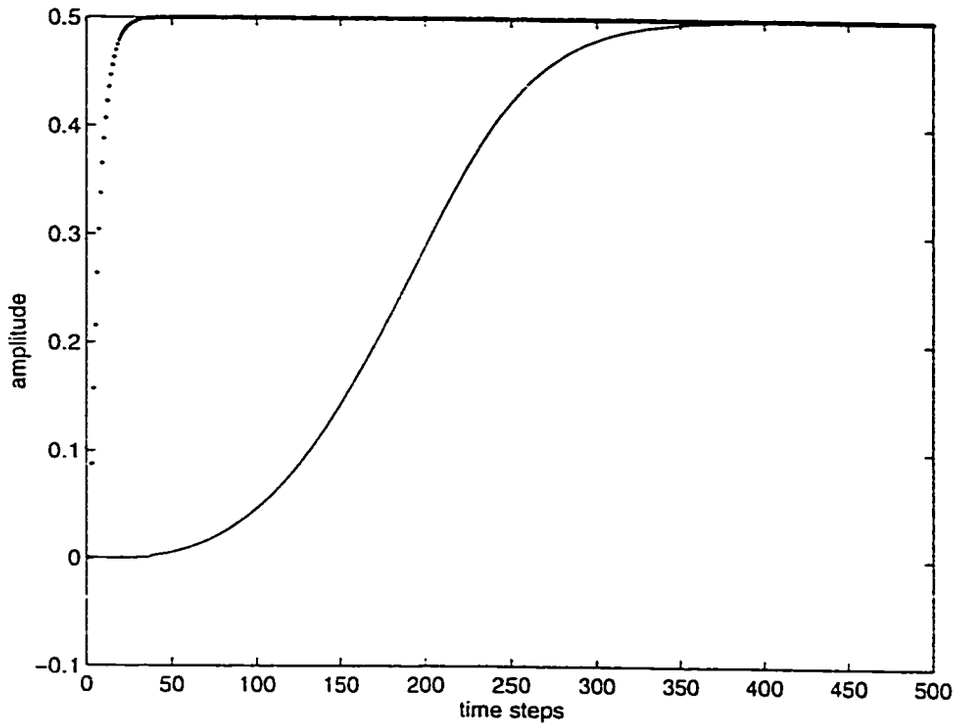


Figure 4.10: Positive step response of the Hammerstein plant (-) and model plant output (..) using untrained FLC

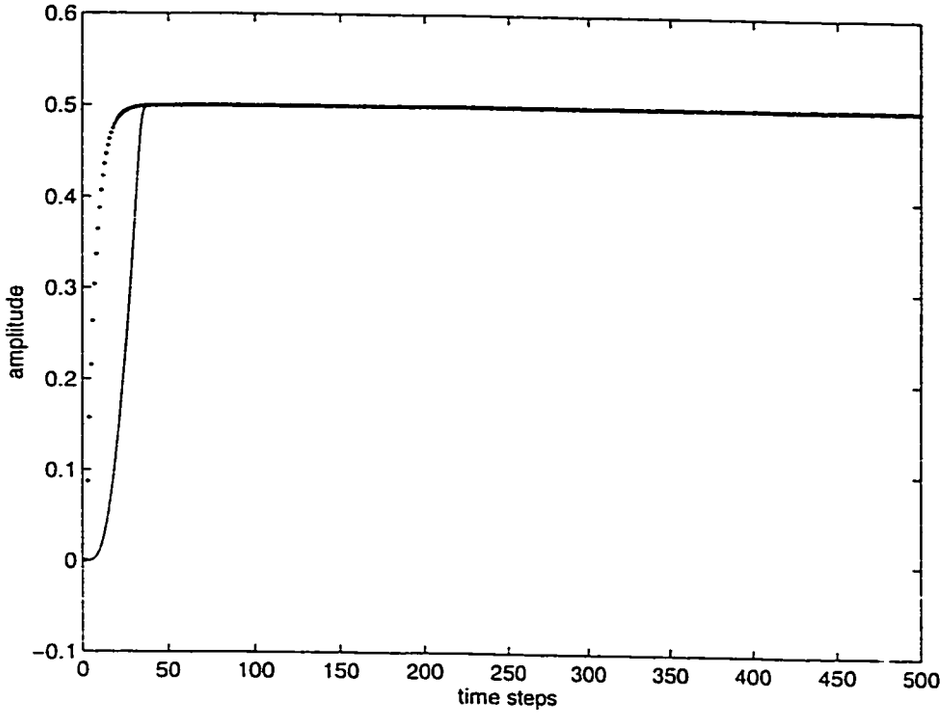


Figure 4.11: Positive step response of the Hammerstein plant (-) and model plant output (..) using trained FLC

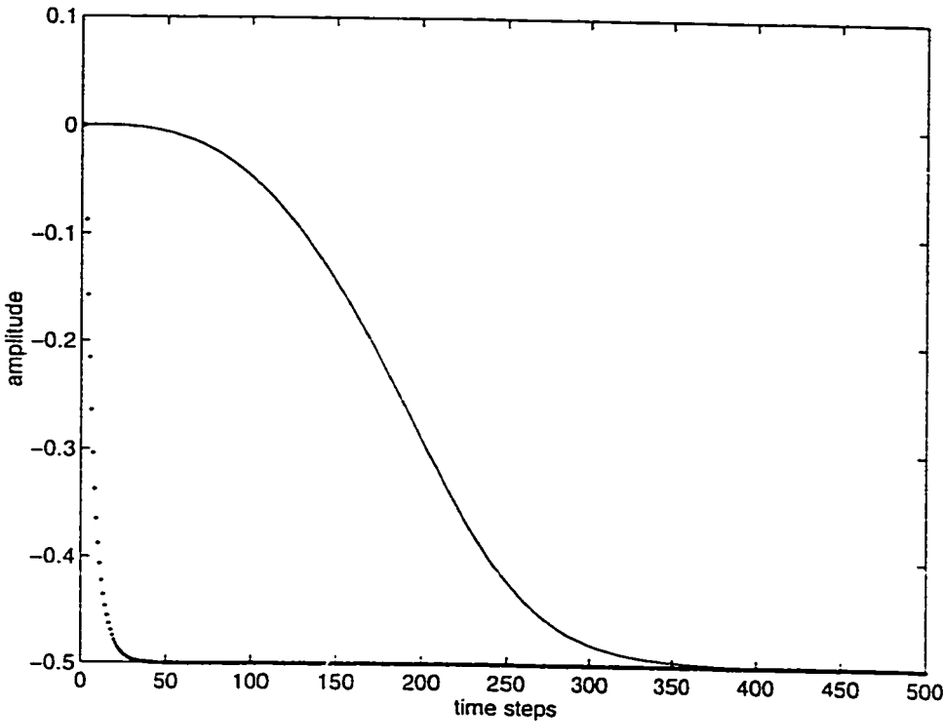


Figure 4.12: Negative step response of the Hammerstein plant (-) and model plant output (..) using untrained FLC

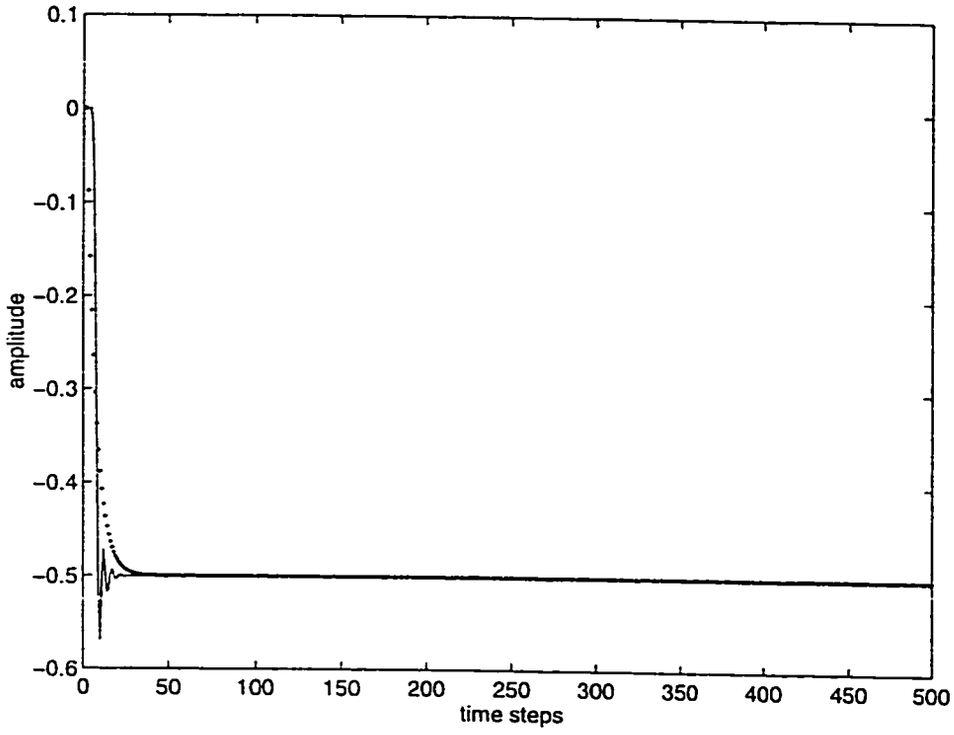


Figure 4.13: Negative step response of the Hammerstein plant (-) and model plant output (..) using trained FLC

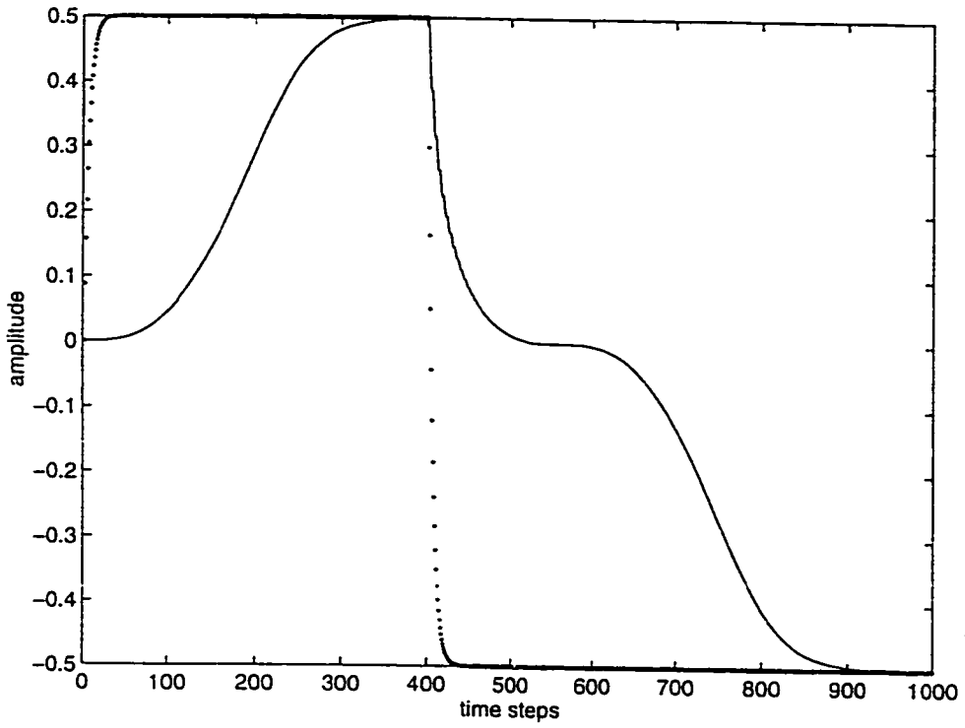


Figure 4.14: Output of the Hammerstein plant using untrained FLC (–) and the model plant output (..) for multiple step inputs

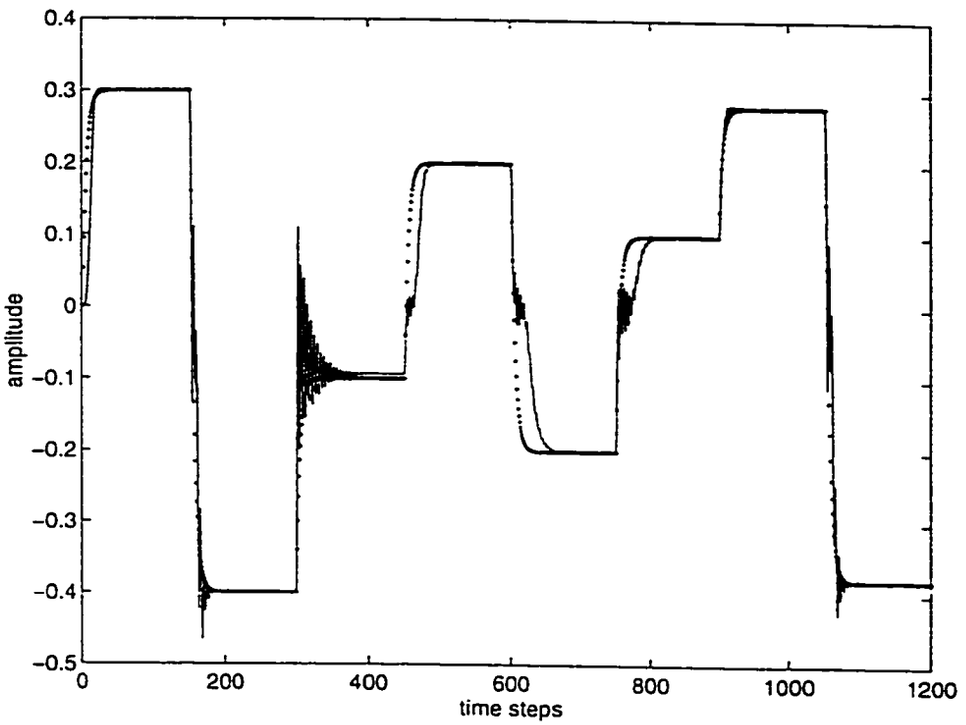


Figure 4.15: Output of the Hammerstein plant using trained FLC (–) and the model plant output (..) for random step inputs

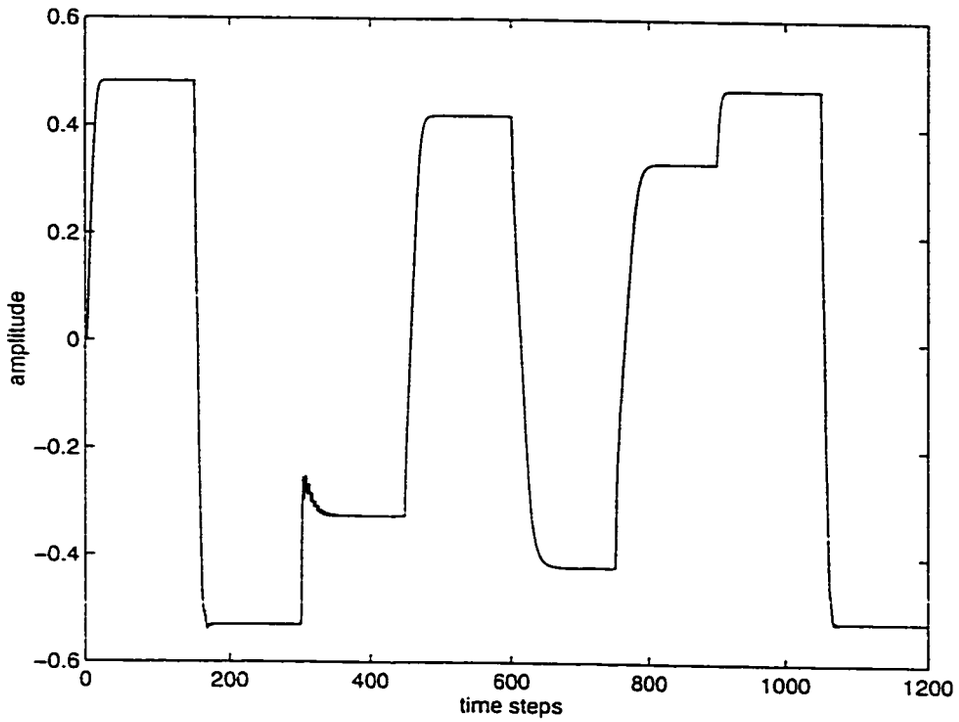


Figure 4.16: Control input signals for Hammerstein plant for random step command

criterion function:

$$J = \Sigma C||y_m - y||^2 + ||D(1 - q^{-1})u||^2$$

where y_m is the reference model output, y is the output of the system and u is the input to the system.

A penalty on the variation in the control input ensures that for a type zero plant, at steady state, the objective function is not affected by the input amplitude. In this example, a plant described by the following equation is considered

$$y(k) - 0.2 * y(k-1) + 0.8737y(k-2) = u(k)^3 + 2.5u(k-1)^3 + u(k-2)^3$$

The plant for the reference model has been the same as in the liquid level system. Figure 4.10 and 4.12 show the positive and negative step responses of the system with the untrained FLC. Note that although the output followed the command. The response is very sluggish. In order to improve the response time, controller training has been necessary. Figures 4.11 and 4.13 show the step responses using trained FLC.

For multiple step commands the untrained system behaved very slowly as shown in Figure 4.14. The response of the system using the trained FLC

is shown in Figure 4.15. In this illustration, the setpoints shown are different than the ones used during the training. In Figure 4.16, the input corresponding to the plant is shown.

A problem occured while training the FLC is that, at times, it became difficult to train the FLC for large transitions. To circumvent the problem, two FLC's are trained which are switched according to the reference command input i.e different FLC's are used for positive and negative transitions.

4.2.1 Robustness of the FLC

Simulation studies have been conducted in order to verify the robustness of the FLC. Table 4.2 shows the range of parameter variation for which the system remained stable. Figure 4.17 shows the step response of the closed loop system with typical parameter variation. Although the performance is effected, the control system remained stable for large range of parameter variations.

Parameter No.	Value	+ve variation %	-ve variation %
1	.2	100	95
2	−.8737	10	95
3	1	100	95
4	2.5	100	95
5	1	100	60

Table 4.2: Variations in parameters of Hammerstein plant for which the system is stable

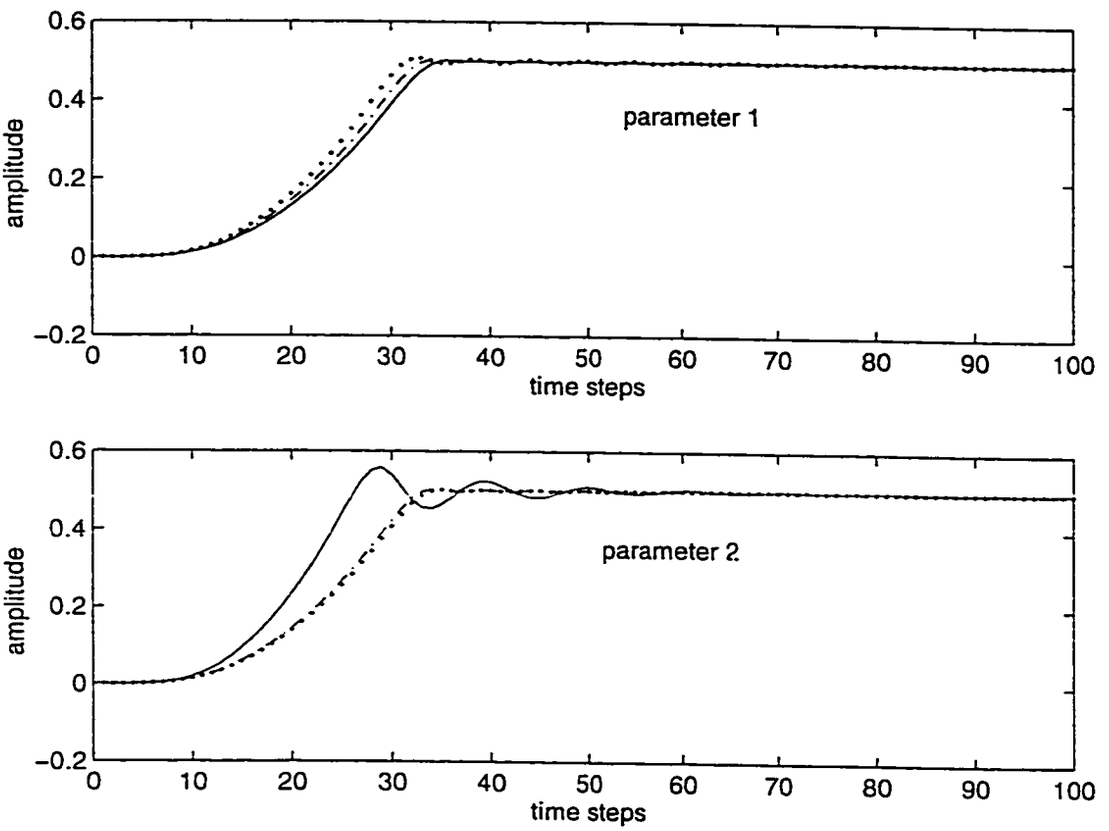


Figure 4.17: Effect of parameter variation on step response of the Hammerstein Plant Nominal(.-). Positive variation (-)((1) 100% (2) 10%) and Negative variation (..)((1) 95 % (2) 95 %)

4.3 Inverted Pendulum

One of the benchmark problems in control systems is the balancing of an inverted pendulum. It is a complex non-linear system which is inherently unstable. When it is displaced from its equilibrium position the intent of the controller is to stabilize it back to the equilibrium point (which as well is unstable). The model for the inverted pendulum is taken as presented in Jang [41] :

$$\ddot{\theta} = \frac{gsin(\theta) + cos(\theta)\frac{(-u - ml\dot{\theta}^2 sin(\theta)}{m_c + m}}{\frac{4}{3} - \frac{mcos^2(\theta)}{m_c + m}}$$

where

g is the acceleration due to gravity.

m_c is the mass of the cart.

m is the mass of the bob.

l is the length of the stick.

θ is the angle which is zero when the pendulum is perpendicular to the cart.

u is the control input.

A first order reference model is taken. The cost function to be minimized is that of Eq 3.14. Initially $C = 1$ and $H^2 = 1e - 3$ while v is 2. Depending

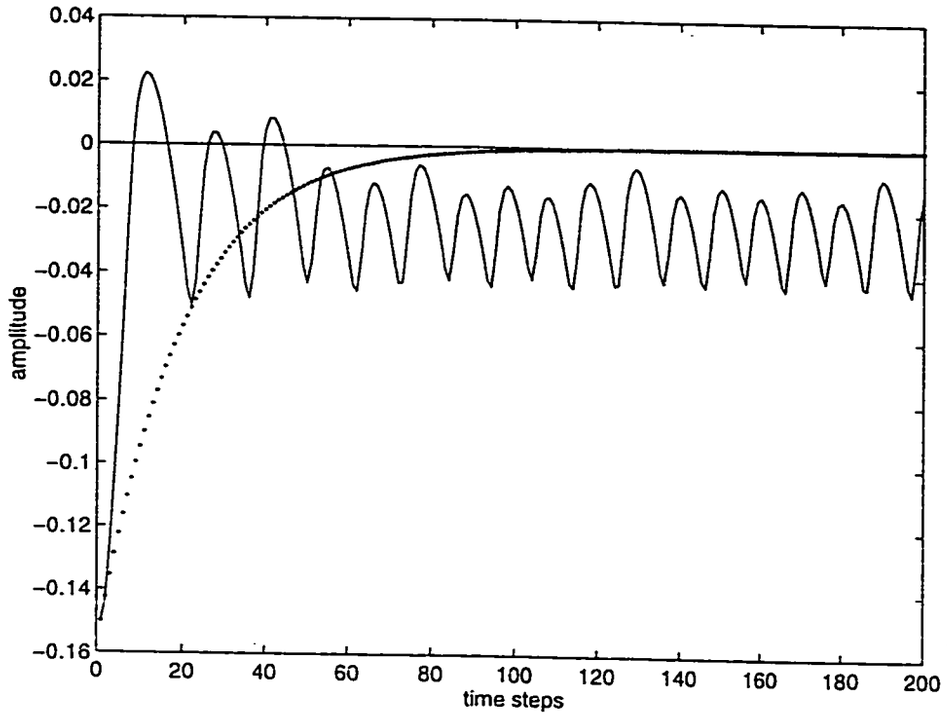


Figure 4.18: Regulation using the untrained FLC: angle of the inverted pendulum starting from a negative initial condition (–) and output of the reference model (..)

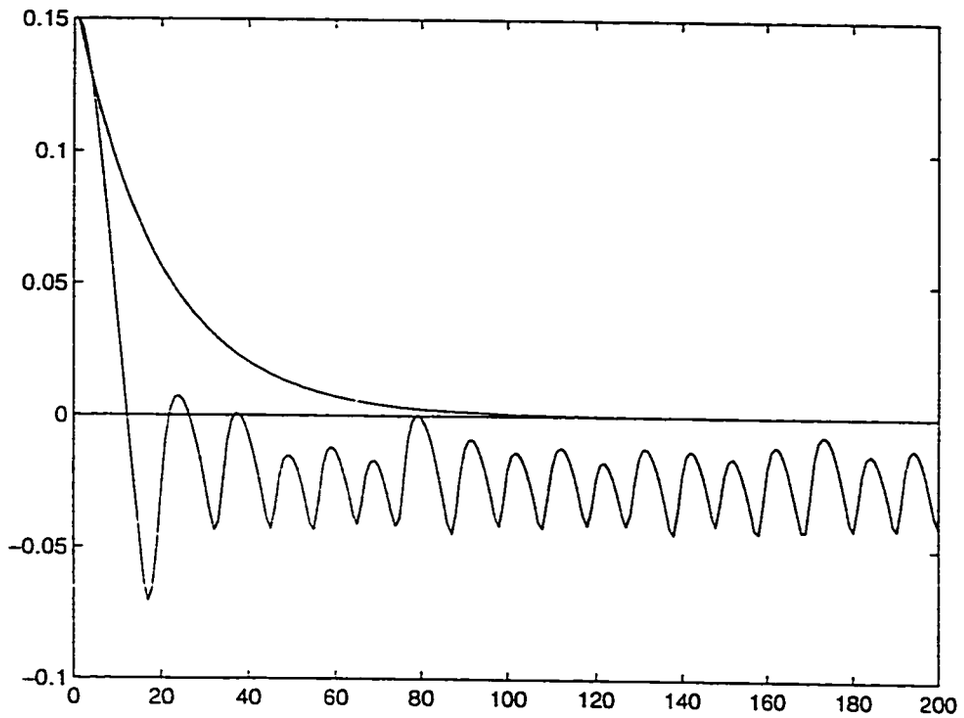


Figure 4.19: Regulation using the untrained FLC: angle of the inverted pendulum starting from a positive initial condition and output of reference model

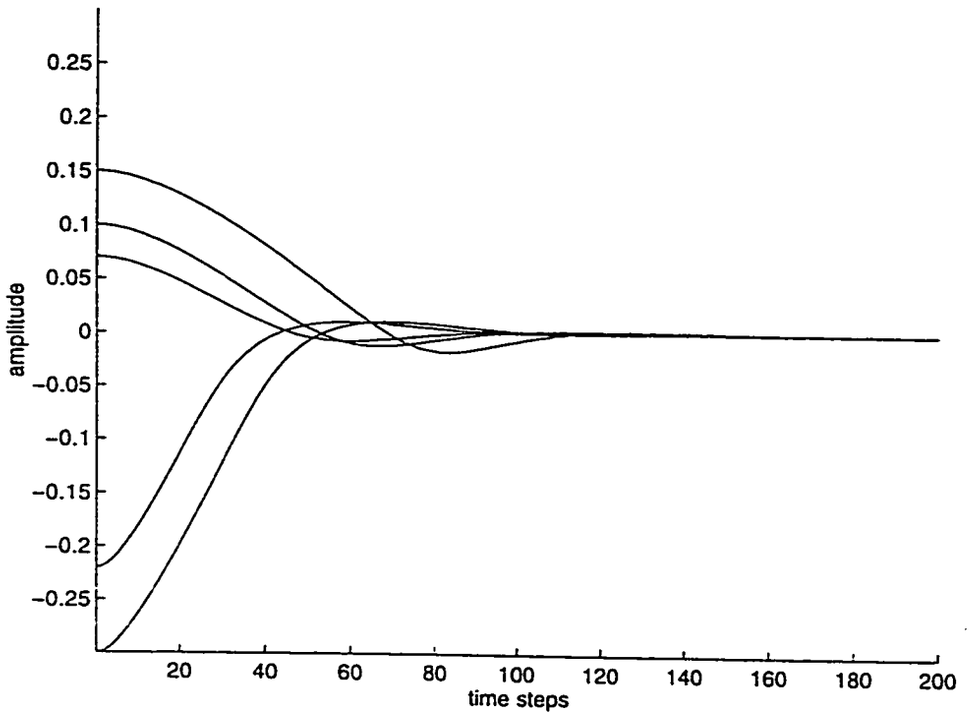


Figure 4.20: Balancing of Inverted pendulum from different initial angular positions (sampling time = .01 sec) using the trained FLC

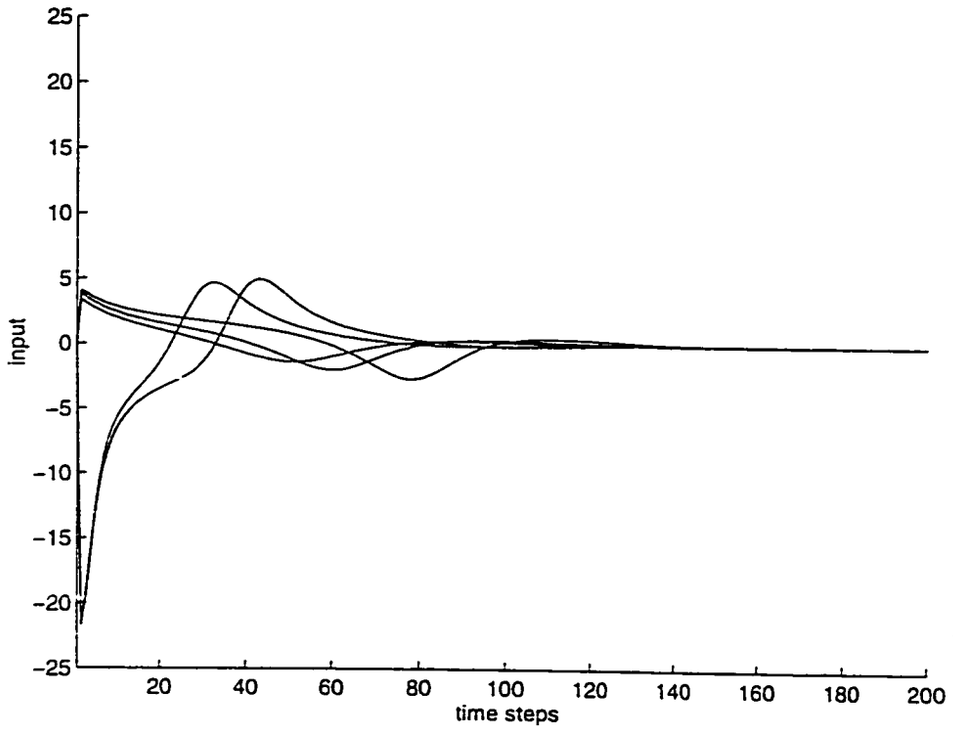


Figure 4.21: Control action driving the pendulum from different initial conditions using the trained FLC

on the convergence these values have been modified later on (details are presented in Section 4.3.5). The sampling time is taken to be 0.01 sec. For the simulation, Fourth order Runge-Kutta method has been used. Figure 4.18 and 4.19 show the output response due to the FLC prior to the proposed training. It can be seen that the initial FLC obtained by trial and error is unable to regulate the angular position of the Inverted pendulum. Although it drives the output towards equilibrium it has been unable to stabilize it satisfactorily. The output θ of the pendulum after training of the FLC is shown in Figure 4.20. The figure shows the angle of the pendulum θ which reaches the zero steady state from different initial conditions. The plant input is shown in Figure 4.21. Training details are presented later in this section.

It should be mentioned here that since this is a regulator problem, the controller has been a pure FLC without the integrator.

4.3.1 Robustness of the FLC

Simulation studies have been conducted to check the sensitivity of the control system against the parameter variation. Simulation is conducted for different

parameter values. Table 4.3 shows the value of parameters for which the simulations have been conducted.

Set no.	m	m_c	l
1(*Nominal*)	0.2	1	0.5
2	0.2	2	0.5
3	0.4	1	0.5
4	0.2	1	1
5	0.4	2	0.5
6	0.4	2	1

Table 4.3: Robustness analysis of IP system (SI units)

The response of the plant for each set of parameters is shown in Figure 4.22

4.3.2 Effect of Disturbance

Effect of disturbance is studied by performing simulations of the control system in the presence of noise. The noise is added at the output of the control system. The types of noise used in the simulation have been:

1. white noise: zero mean, Gaussian with variance 1.6e-5

2. impulse noise: absolute maximum amplitude .01

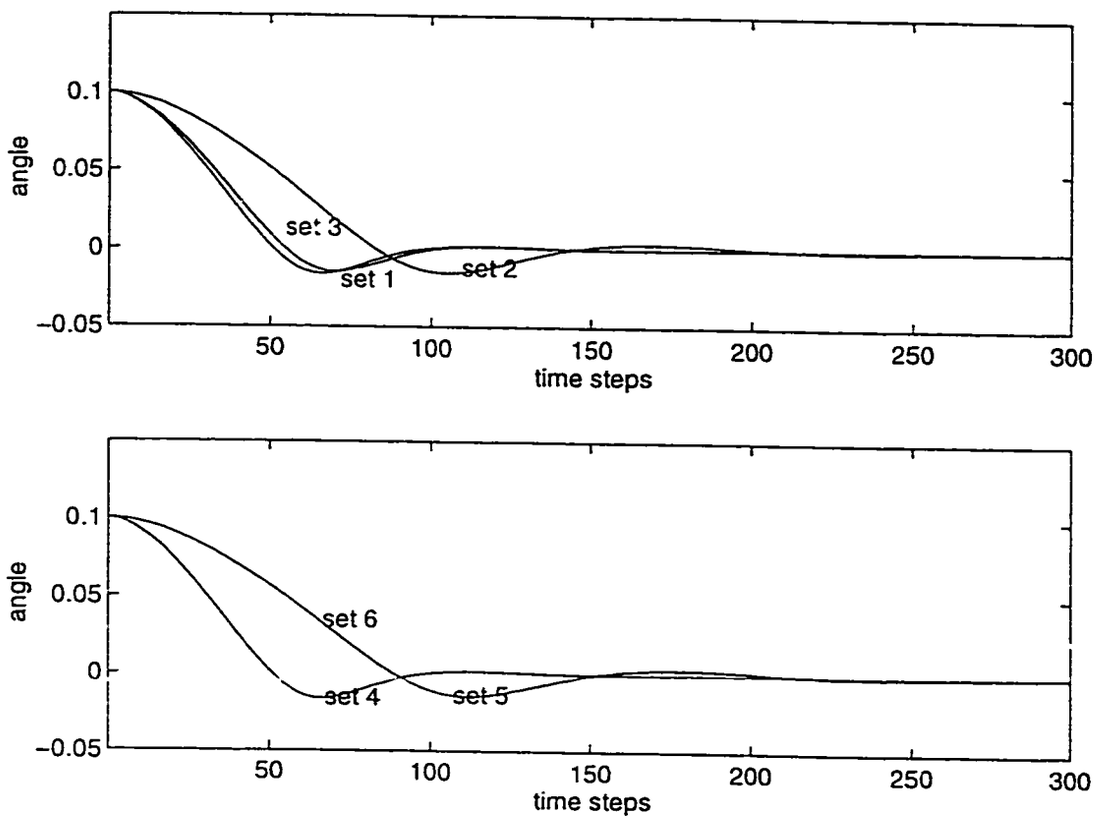


Figure 4.22: Effect of parameter variation on the inverted pendulum

Figure 4.23 shows the effect of noise on the angular position of the Inverted pendulum system. It can be observed that the responses have been satisfactory. In the presence of random disturbance, the output remained bounded. Whereas in the presence of impulse noise, the output quickly approached the desired values.

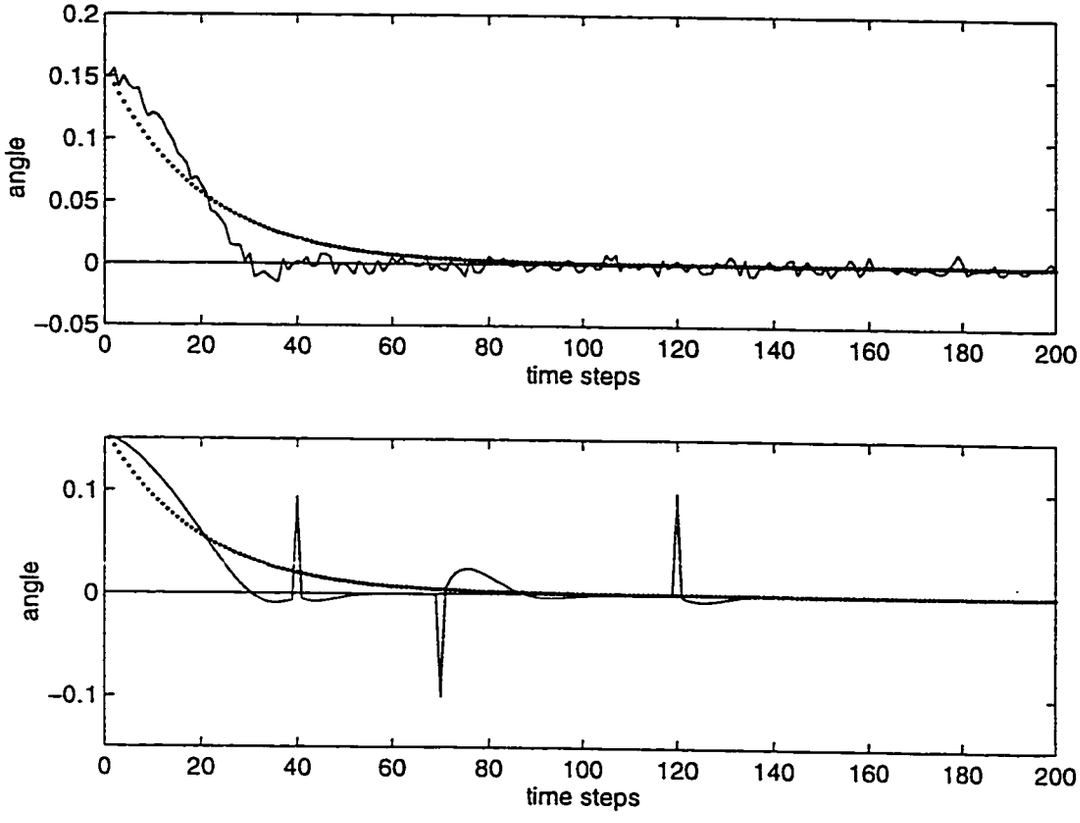


Figure 4.23: Effect of noise on the IP system (i) white noise added (ii) impulse noise added

4.3.3 Computation time

The proposed scheme is computationally intensive. For the case of inverted pendulum, it took 70 hours of training time on a 166 MHz Pentium PC. This time constitutes 70% of the computation time while 30% time is consumed in experimentation for adjustment of the training parameters.

4.3.4 Details of parameters tuned

For the FLC of the inverted pendulum case, the parameters used are shown in the Table 4.4.

Parameter	No.
No. of Inputs	2
No. of Input membership functions for each input	3
No. of centroids of Input membership functions	6
No. of variances of Input membership functions	6
No. of output membership functions	9
Input scaling factor k_u	1
Output scaling factor k_y	1
Error scaling factor k_e	1
Δ Error scaling factor k_{de}	1

Table 4.4: No. of parameters used in the FLC for the inverted pendulum

4.3.5 Training details

In order to provide an idea about the training process, details of a sample training session are presented in this section. Although the entire training procedure may vary for different problem and different user, the following shows a typical training session on the inverted pendulum problem.

The training is started from the FLC obtained by trial and error. Although such an FLC has been able to drive the Inverted pendulum towards equilibrium but the response has been highly oscillatory. Table 4.4 shows the sets of parameters which are used in this guided training. The convergence of the objective function is shown in Figure 4.24 and 4.25.

Initially a very small penalty has been used on the control input. Learning rate for each parameter is adjusted by trial and error. As soon as it appeared that the rate of convergence became slow, the learning rates have been increased. This strategy is followed till Phase 2(viii) (shown in Table 4.5). In the table the entries η_{y^l},η_{μ^l},η_{σ^l} and η_{bias} are the multiplication factors that have been multiplied to η in order to get the respective learning rate for each parameter.

At the end of Phase 2(viii) (as shown in Figures 4.26 and 4.27) it has

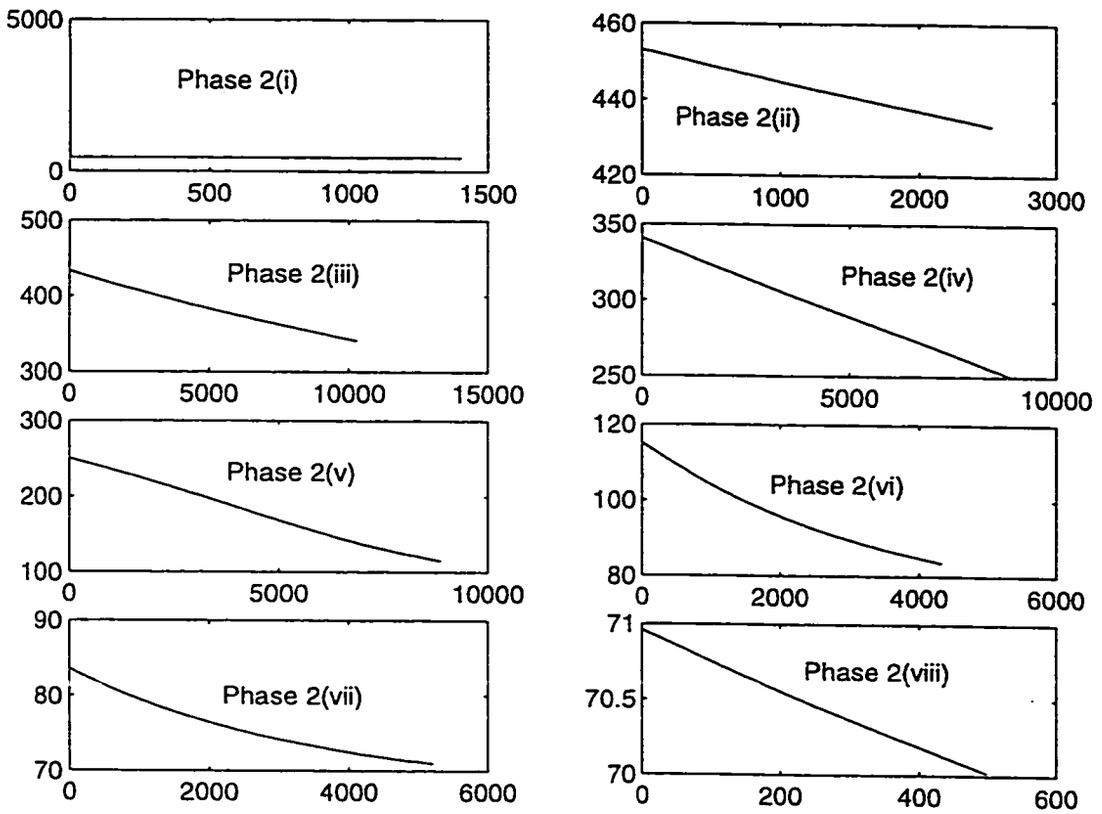


Figure 4.24: Convergence of Objective function for Phase 2 of the training

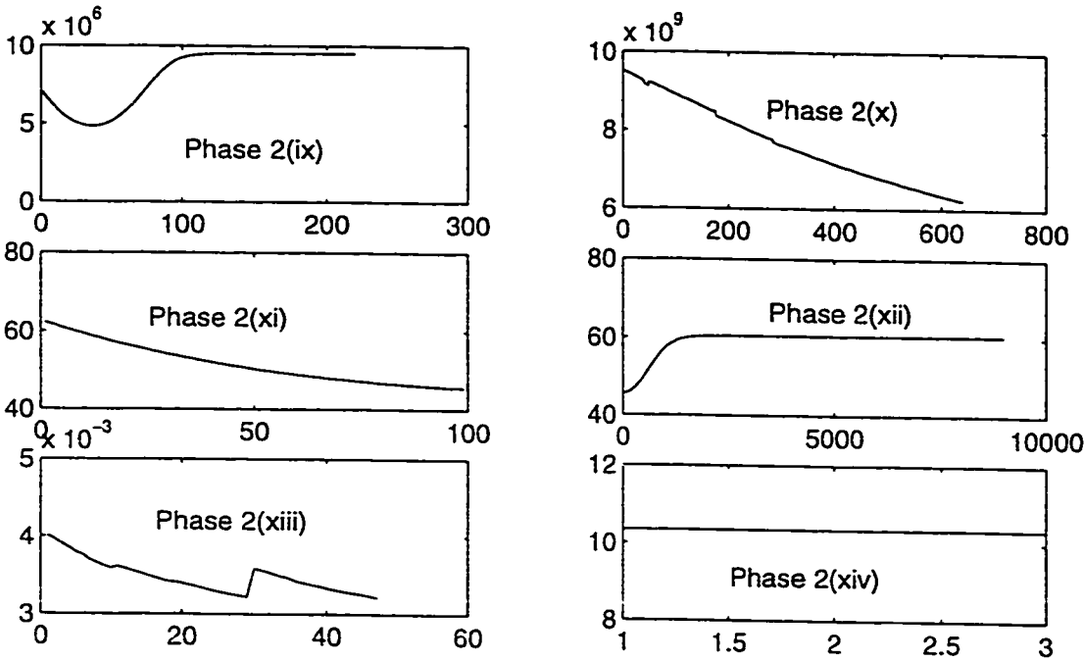


Figure 4.25: Convergence of Objective function for Phase 2 of the training

Phase	η_{y^l}	η_{μ^l}	η_{σ^l}	η_{bias}	η	Iterations	Error Penalty	Input Penalty
$2(i)$	$5e-1$	1e-1	1e-1	0	24e-6	1399	1	$(1e\text{-}3)t^2$
$2(ii)$	$5e-1$	1e-1	1e-1	0	36e-6	2529	1	$(1e\text{-}3)t^2$
$2(iii)$	$5e-1$	1e-1	1e-1	0	60e-6	10289	1	$(1e\text{-}3)t^2$
$2(iv)$	$5e-1$	1e-1	1e-1	0	130e-6	8879	1	$(1e\text{-}3)t^2$
$2(v)$	$5e-1$	1e-1	1e-1	0	130e-6	8839	1	$(1e\text{-}3)t^2$
$2(vi)$	$5e-1$	1e-1	1e-1	0	170e-6	4319	1	$(1e\text{-}3)t^2$
$2(vii)$	$5e-1$	1e-1	1e-1	0	220e-6	5199	1	$(1e\text{-}3)t^2$
$2(viii)$	$5e-1$	1e-1	1e-1	0	440e-6	499	1	$(1e\text{-}3)t^2$
$2(ix)$	0	0	0	10	500e-9	219	0	$10t^2$
$2(x)$	$5e-1$	1e-1	1e-1	0	500e-14	639	0	$(100t)^2$
$2(xi)$	$5e-1$	1e-1	1e-1	0	5e-3	99	0	$(.01t)^2$
$2(xii)$	0	0	0	2	5e-3	8979	0	$(.01t)^2$
$2(xiii)$	0	0	2	200	5e-1	47	1	0
$2(xiv)$	0	1e-3	0	0	5e-1	3	1	0

Table 4.5: The 2nd phase training of the Inverted pendulum

been observed that the angular position attain non-zero steady state value. With further training the objective function increased at this point, no matter what are the values of the learning rates. However, it has been possible to attain zero steady state angular position by modifying the criterion function by increasing the penalty on the input amplitude.

In this step, the training of conventional parameters has been disabled and using additional penalty on control input, *bias* is trained. This involved some effort consisting of adjusting the learning rate and observing the convergence of the objective function. Thus parameters of FLC have been trained by varying the penalty on u during this phase.

In the end, further penalty on u no longer affected the objective function, hence training has been done by penalizing only the error. The final response is shown in the Figure 4.26 and 4.27 which shows that a good response has been achieved. In this training scaling factors have been unity except k_y that is zero.

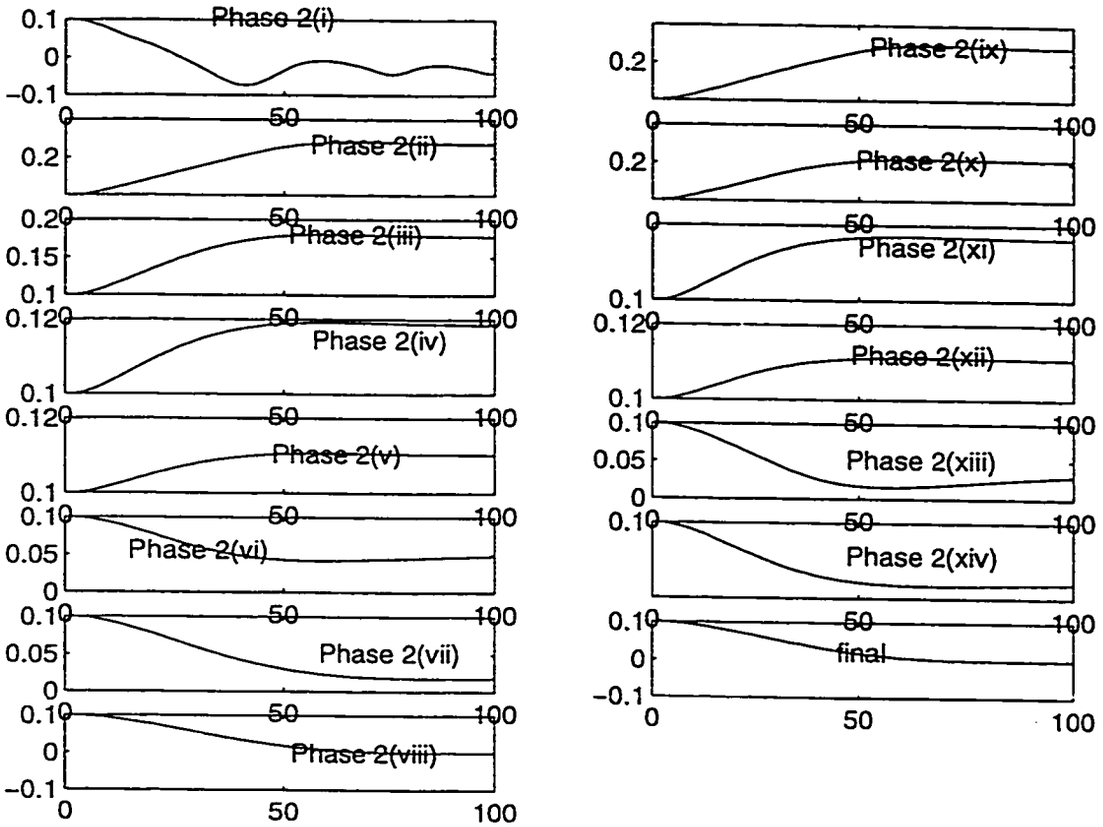


Figure 4.26: The angular position of the Inverted pendulum at the start of each training phase and the final response

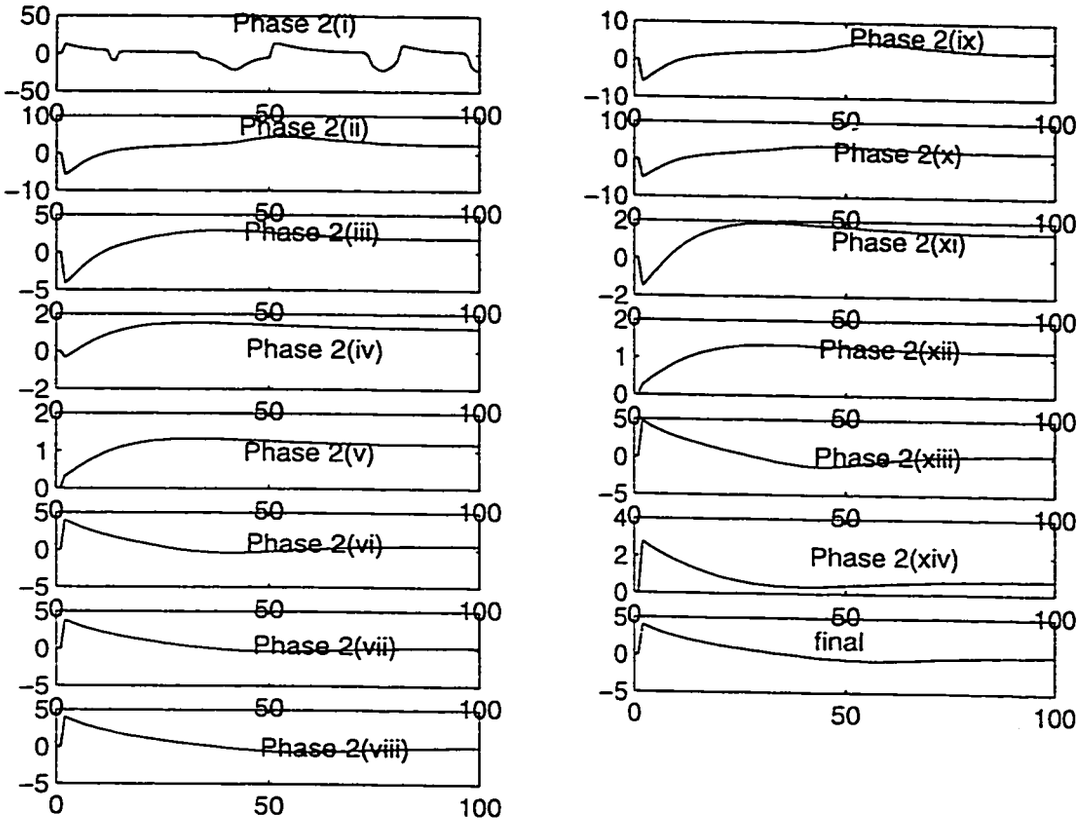


Figure 4.27: Control input applied to the Inverted pendulum at the start of each training phase and the final response

4.3.6 Behavior of Parameters

In order to track the behavior of the parameters during the training, training data of the centroid 1 and centroid 2 (of the input membership function of error) is recorded. This is shown in the Figure 4.28. It is observed that the parameters converge to constant towards the end of the training.

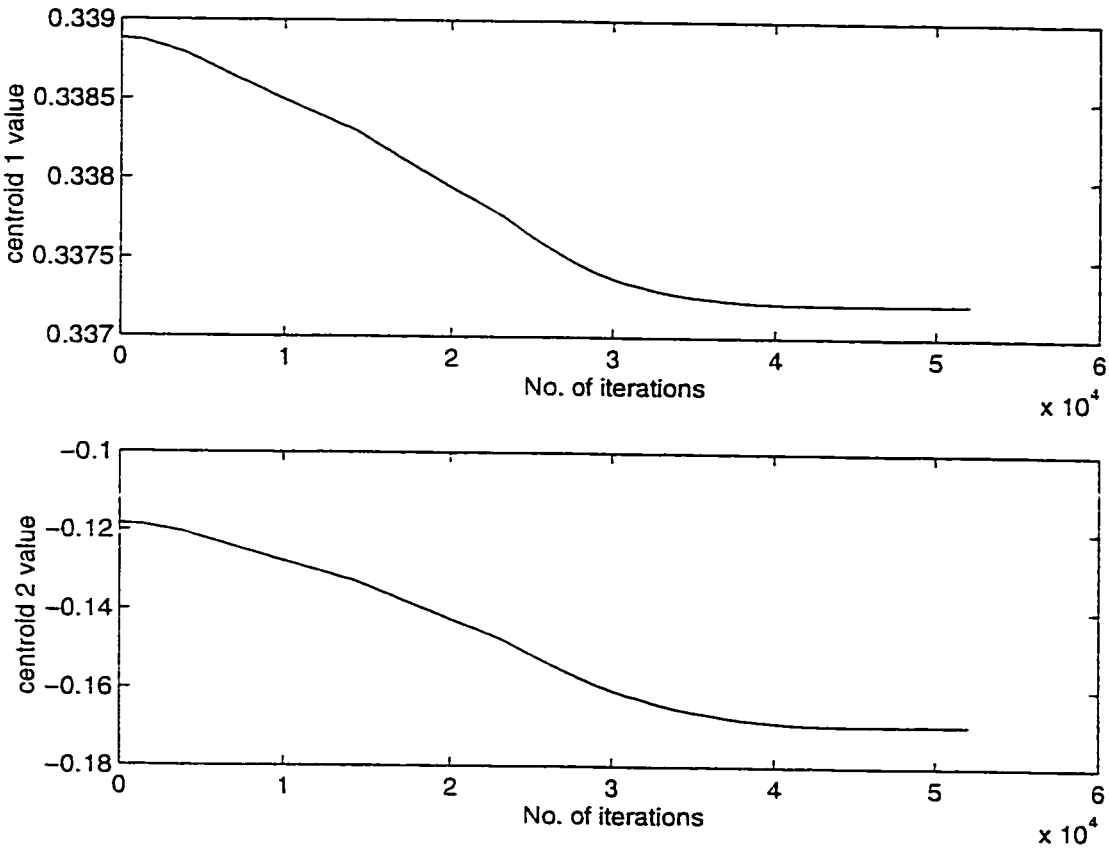


Figure 4.28: Convergence of FLC parameter values for different sets of training

4.4 Comparison with Existing schemes

As described in the second chapter that there have been a number of training schemes for the FLC. However the proposed scheme has the following advantages:

1. The scheme is well structured as compared to the most trial and error schemes.

2. The inverse model need not be identified, this extends its applicability to non-minimum phase plants.

3. There is no limitation on the structure of the plant except that it has to be first order differentiable.

4. For complex control systems, the scheme is easily extendible.

In order to assess the effectiveness of the proposed scheme, simulation results are compared with one of the recent FLC training schemes i.e ANFIS (Adaptive Neuro Fuzzy Inference System) as presented by Jang in 1992 [40][41]. This scheme was also applied by the author to the Inverted Pendulum system. Figure 4.29 shows a quantitative comparison of the ANFIS scheme with the proposed scheme. It could be seen that settling time of θ

for the ANFIS scheme is smaller than that of our scheme but is comparable. However in the comparison of the input force amplitude our scheme exhibits better results. In practice, there is a tradeoff between these two attributes. Larger the magnitude of the input force, smaller will be the settling time. Hence, both the schemes show equivalent results. Using different penalty on the input force, the same result as that of ANFIS scheme can be achieved. But it depends on the preference of the system designer to choose the specification in order to meet practical considerations.

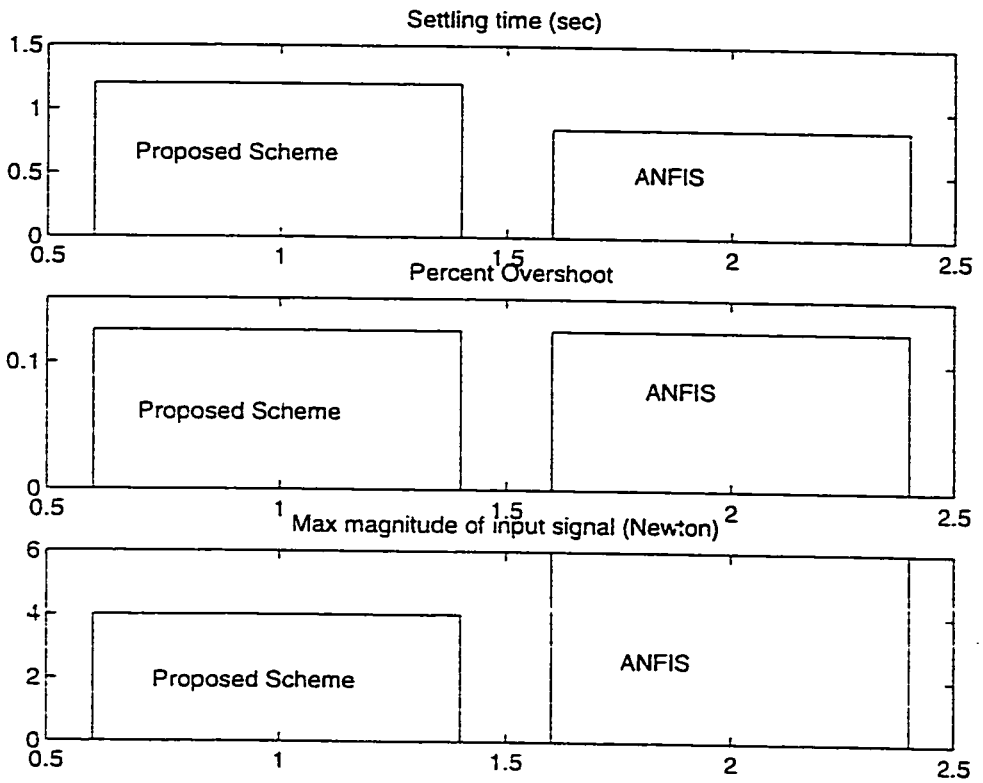


Figure 4.29: Comparison of the ANFIS and proposed scheme

Chapter 5

Conclusions and Comments

A general technique for the optimal training of FLC for non-linear plants is presented. Simulations for the servo and regulator problems are done to validate the scheme. The method is easily extendible to control systems having complex structures.

The method is computationally intensive but it has many advantages. Its salient features are the systematic method of training and applicability to complex control systems. A variety of objective functions could be selected by choosing different values of Penalty functions, including the time weighted optimal control problem.

5.1 Future Directions

- Work can be extended towards simplification of the computations.
- The proposed design technique is completely off-line. It can easily be extended to an on-line adaptive scheme. Such extension requires considerable simplification of the parameter adjustment procedure.
- The applicability of this method can be tested on real laboratory scale engineering systems.

References

[1] L. A. Zadeh " Fuzzy sets" , Information and control vol. 8, pp 338-353, 1965

[2] P. J. King and E. H. Mamdani " The application of Fuzzy control systems to Industrial Processes" Automatica, vol 13, no 3, pp 235-242, 1977

[3] E. H. Mamdani " Application of fuzzy algorithms for control of simple dynamic plant" Proceedings of the IEEE, vol 121, no 12 pp 1585-1588, 1976

[4] T. J. Procyk and E. H. Mamdani " A Linguistic self organizing process controller" Automatica, vol 15, pp 15-30, 1979

[5] Bart Kosko "Fuzzy systems as universal approximators" IEEE transactions on computers, vol 43, no 11, 1994

[6] Daniel G. Schwartz, Harold W. Lewis III and Yoshinori Ezawa " Applications of Fuzzy sets and approximate reasoning" Proceedings of the IEEE, vol 82, no 4, April 1994

[7] Chuen Chien Lee, " Fuzzy Logic in control systems Part I- Part II " IEEE transactions on systems, man and cybernetics, vol 20, no2 , Mar/Apr 1990

[8] Piero. P. Bonisso, Vivek Badami, Kenneth H. Chiang, Pratap S. Khedkar, Kenneth W. Marcelle, Michael J. Schutten " Industrial Applications of

Fuzzy Logic at General Electric" vol 83, no3, Mar 1995

[9] Tomohiro Takagi and Michio Sugeno " Fuzzy identification of systems and its applications to Modeling and control" IEEE transactions on systems, man and cybernetics, vol SMC-15, no 1, Jan-Feb 1988

[10] M.M. Gupta and J. Qi " Fusion of Fuzzy Logic and neural networks with applications to decision and control problem" Proceedings of American control conference, pp 30-31, 1991

[11] J. Keller and Hussein Tahani " Back propagation neural networks for Fuzzy Logic" Information sciences, vol 62 pp 205-221, 1992

[12] Ching Teng Lin and C. S. George Lee "Neural network based Fuzzy Logic control in decision systems" IEEE transactions on computers, vol 40, no 12, Dec 1991

[13] Ching Teng Lin and C. S. George Lee " Reinforcement structure/ parameter learning of neural network based Fuzzy Logic Control Systems" IEEE transactions on Fuzzy systems, vol 2, no 1, Feb 1994

[14] Junhong Nie and D. A. Linkens " FCMAC : A fuzzified cerebellar Model articulation controller with self organizing capacity" Automatica, vol 30, no 4, pp 655-664, 1994

[15] Dianhui Wang , Tianyou Chai and Linhua Xia " Adaptive Fuzzy neural

controller design" Proceedings of the American control conference, Seattle, Washington, June 1995, pp 4258-4262

[16] T. K. Yin, Tang-Kai and C. S. George Lee "Fuzzy Model Reference Adaptive control" IEEE transactions on systems, man and cybernetics, vol. 25,no. 12, Dec 1995

[17] Diahee Park, Abraham Kandel and Gideaon Langholz " Genetic-Based New Fuzzy Reasoning models with applications to Fuzzy control " IEEE transactions on systems, Man and cybernetics, vol. 24, no. 1, Jan 1994

[18] Derrick . H. Nguyen and Bernard Widrow " Neural networks for self learning control systems" IEEE control systems magazine pp 18-23 Apr 1990

[19] J. R. Layne "Fuzzy Model Reference Learning control for cargo ship steering " IEEE control systems magazine , vol 13, no 6, pp23-34, Dec 1993

[20] Kazuo Tanaka, Manabu Sano and Hiroyuki Watanabe " Modeling and control of Carbon Monoxide concentration using a neuro fuzzy technique" IEEE transactions on Fuzzy systems, vol 3 no 3, Aug 1995

[21] Waihon A. Kwong and Kevin M. Passino " Dynamically focussed Fuzzy learning control" IEEE transactions on systems, man and cybernetics: Part B vol 26, no 1 , Feb 1996

[22] Thomas Hessburg, and Masayoshi Tomizuka "Model Reference Adap-

tive Fuzzy Logic control for vehicle guidance" Proceedings of the American control conference, Seattle, Washington, pp 2287-2291, June 1995

[23] Jeffry T. Spooner and Kevin M. Passino " Stable direct adaptive control using fuzzy systems and neural networks" Proceedings of the 34th conference on decision and control, New Orleans, LA, pp 249-254, Dec 1995

[24] Waihon A. Kwong, Devin M. Passino, Eric G. Lankonen and Stephen Yurkovich "Expert Supervision of Fuzzy Learning systems for Fault tolerant Aircraft control " Proceedings of the IEEE , vol 83, no 3 , Mar 95

[25] D. A. Linkens and H. O. Nyongesa "Genetic Algorithms for fuzzy control Part I off-line system development and application" IEE proceedings of control theory and applications, vol 142, no 3, May 1995

[26] M. S. Ahmed " Block partial derivative and its application to neural-net-based direct-model-reference adaptive control" IEE Proceedings- Control Theory and Applications, vol. 141, no. 5, Sep 1994

[27] Bor-Sen Chen, Ching Hsiang Lee and Yeong-Chan Chang *et al.*" H infinity tracking design of uncertain non-linear SISO systems: Adaptive fuzzy approach" IEEE transactions on Fuzzy systems, vol 4, no 1, Feb 96

[28] Ching-Geng Lin and Ya Ching Lu " A neural Fuzzy system with Linguistic teaching signals" IEEE transactions on Fuzzy systems, vol. 3, no. 2,

May 1995

[29] Hamid R. Berenji and Pratap Khedkar " Learning and Tuning Fuzzy Logic controllers through reinforcements" IEEE transactions on neural networks. vol. 3. no. 5. Sep 1992

[30] V. G. Mougdel *et al.*" Fuzzy learning control for a flexible link robot" Proceedings of the 1994 American control conference 1994 Green valley, AZ, pp 563-567

[31] M. S. Ahmed "BPD Computation and Model Reference Adaptive Control of Hammerstein Plants,"IEE Proceedings-Control Theory and Applications. vol. 142. no.5. 1995

[32] Yoshiharu Ohtani and Toshio Yoshimura " Fuzzy control of a manipulator using the concept of sliding mode" International journal of systems science. vol 27. no 2. pp 179-186. 1996

[33] M. Ayoubi " Neuro-Fuzzy structure for rule generation and application in the fault diagnosis of technical processes" Proceedings of the American control conference. seattle. WA. pp 2757-2761. June 1995

[34] Li-Xin Wang " Adaptive Fuzzy systems and control: Design and stability analysis" Prentice Hall. 1994

[35] Payman Arabshahi. Jai J. Choi. Robert J. Marks II. Thomas P. Candell

" Fuzzy parameter adaptation in optimization" IEEE computational science and engineering, pp 57-65, Spring 1996

[36] C. L. Karr and Edward J. Gentry "Fuzzy control of pH using Genetic Algorithms" IEEE transactions on Fuzzy systems, vol 1, nol. 1, pp 47-53, Feb 1993

[37] A. Trebi Ollennu and B. A. White " Robust output tracking for MIMO non-linear systems : an adaptive Fuzzy approach" Proceedings of the 34th conference on Decision and control. New Orleans, LA, Dec 1995, pp 273-279

[38] K. Krishnakumar, P. Gonsalves, A. Satyadas and G. Zacharias " Hybrid fuzzy Logic Flight controller synthesis via pilot modeling " Journal of Guidance, control and dynamics,, vol. 18, no. 5. Sep-Oct 1995

[39] Watanabe Keigo. Jun Tang, Masatoshi. Shinji Koga and Toshio Fukuda " A Fuzzy Gaussian neural network and its application to mobile robot control" IEEE transactions on control systems technology, vol 4, no. 2, Mar 1996

[40] Jyh-Shing R. Jang " Self learning Fuzzy controllers based on temporal back propagation" IEEE transactions on neural networks, vol 3, no. 5. Sep 1992, pp 715-723

[41] Jyh-shing Roger Jang and Chuen Tsai sun " Neuro Fuzzy modeling and

representations by error propagation, " in Parall Distributed Processing: Explorations in the Microstructure of Cognition, D.E. Rumelhart and James L. McClelland. Eds.. vol. 1. Cambridge, MA: MIT Press, 1986, ch. 8, pp. 318-362

[50] Webrose. P. " Backpropagation through time, What it does and How to do it" Proc. of IEEE. vol 78. No 10. Oct 1990, pp1550-1560

[51] Narendra. K. and K. Parthasarthy " Gradient method for the optimization of dynamical systems containing neural networks ", IEEE transactions on neural networks. vol 2. Mar 1991, pp 252-262

[52] Narendra. K. and K. Parthasarthy "Neural Networks is control systems" Proc. 31st IEEE conf. Dec. Control, Tucson. Arizona, Dec 1992. pp1-5

[53] Astrom. K. J and Bjorn Wittenmark "Adaptive Control" Addison-Wesley Publishing Company. Inc. 1995